Investigating automated chemical evolution of oil-in-water droplets

Juan Manuel Parrilla Gutiérrez

A thesis submitted to the University of Glasgow

for the degree of Doctor of Philosophy

School of Chemistry

College of Science and Engineering

September 2016

This thesis is dedicated to: my family for all their support,
my girlfriend, for all the adventures and happiness,
and our dog *Cassie*, for taking me out for a walk when I needed it.

"I would have written a shorter *thesis*, but I did not have the time."
Adapted from Blaise Pascal

Acknowledgements

This project was carried out between October 2012 and September 2016 in the Department of Chemistry at the University of Glasgow. Many people helped me during this project, and in particular I would like to thank to:

Prof. Leroy Cronin for giving me the opportunity to work in his group and for his support, encouragement and supervision at all times during these years.

All the past and present members of the **Cronin group** who I met during all these years. I do no want to say names, because it is not nice when you check someone else's thesis and your name is not there. Therefore, if you expected to see your name here, it is here, I promise.

Publications

The following articles were published as a result of work undertaken over the course of this PhD programme:

1. *Evolution of oil droplets in a chemorobotic platform.*

   J.M.P Gutiérrez, T. Hinkley, J.W. Taylor, K. Yanev and L. Cronin,

   Nature Communications **5**(5571), (2014)

2. *Towards heterotic computing with droplets in a fully automated droplet-maker platform.*

   A. Henson, J.M.P Gutiérrez, T. Hinkley, S. Tsuda and L. Cronin,

   Philosphical Transactions of the Royal Society A **373**(2046), (2015)

3. *Embodied evolution of droplets in 3D-printed devices with programmable environments.*

   J.M.P Gutiérrez, S. Tsuda, J. Grizou, J.W. Taylor, A. Henson, and L. Cronin,

   Manuscript in preparation

Abstract

One of the main unresolved questions in science is how non-living matter became alive in a process known as *abiognesis*, which aims to explain how from a *primordial soup* scenario containing simple molecules, by following a "bottom up" approach, complex biomolecules emerged forming the first living system, known as a *protocell*. A protocell is defined by the interplay of three sub-systems which are considered requirements for life: information molecules, metabolism, and compartmentalization. This thesis investigates the role of compartmentalization during the emergence of life, and how simple membrane aggregates could evolve into entities that were able to develop "life-like" behaviours, and in particular how such evolution could happen without the presence of information molecules.

Our ultimate objective is to create an autonomous evolvable system, and in order tp do so we will try to *engineer life* following a "top-down" approach, where an initial platform capable of evolving chemistry will be constructed, but the chemistry being dependent on the robotic adjunct, and how then this platform can be de-constructed in iterative operations until it is fully disconnected from the evolvable system, the system then being inherently autonomous.

The first project of this thesis describes how the initial platform was designed and built. The platform was based on the model of a standard liquid handling robot, with the main difference with respect to other similar robots being that we used a 3D-printer in order to prototype the robot and build its main equipment, like a liquid dispensing system, tool movement mechanism, and washing procedures. The robot was able to mix different components and create populations of droplets in a Petri dish filled with aqueous phase. The Petri dish was then observed by a camera, which analysed the behaviours described by the droplets and fed this information back to the robot. Using this loop, the robot was then able to implement an evolutionary algorithm, where populations of droplets were evolved towards defined life-like behaviours.

The second project of this thesis aimed to remove as many mechanical parts as possible from the robot while keeping the evolvable chemistry intact. In order to do so, we encapsulated the functionalities of the previous liquid handling robot into a single monolithic 3D-printed device. This device was able to mix different components, generate populations of droplets in an aqueous phase, and was also equipped with a camera in order to analyse the experiments. Moreover, because the full fabrication process of the devices happened in a 3D-printer, we were also able to alter its experimental arena by adding different obstacles where to evolve the droplets, enabling us to study how environmental changes can shape evolution. By doing so, we were able to embody evolutionary characteristics into our device, removing constraints from the physical platform, and taking one step forward to a possible autonomous evolvable system.

Contents

1 Introduction **1**
- 1.1 From the origin of life to protocells . . . 1
  - 1.1.1 Trying to define *life* . . . 1
  - 1.1.2 The origin of life . . . 2
  - 1.1.3 Autopoiesis and Chemoton theoretical models . . . 7
  - 1.1.4 Protocells . . . 10
- 1.2 Surfactant chemistry primer . . . 14
  - 1.2.1 Chemical polarity . . . 14
  - 1.2.2 Amphiphiles . . . 17
  - 1.2.3 Surface tension . . . 18
  - 1.2.4 Surfactant Classification . . . 18
  - 1.2.5 Aggregates . . . 19
  - 1.2.6 Surfactants and the origin of life . . . 21
- 1.3 From protocells to oil droplets . . . 23
  - 1.3.1 Movement . . . 25
  - 1.3.2 Division . . . 27
  - 1.3.3 "Multicellular" . . . 29
  - 1.3.4 Other *life-like* behaviours . . . 31
  - 1.3.5 Other droplet based research . . . 32
- 1.4 Evolving droplet experimentation . . . 33
  - 1.4.1 Chemical evolution . . . 34
  - 1.4.2 Genetic algorithms . . . 35
  - 1.4.3 Evolutionary algorithms applications . . . 39
  - 1.4.4 The role of the environment during evolution . . . 41
  - 1.4.5 Open-ended evolution . . . 45
- 1.5 Automating droplet experimentation . . . 46
  - 1.5.1 Liquid handling robots . . . 46
  - 1.5.2 Micro and milli-fluidic devices . . . 49
  - 1.5.3 3D-printed *reactionware* . . . 52
  - 1.5.4 Droplets as artificial life *lifeforms* . . . 53

2 Aims **56**

3 Dropbot platform **58**
- 3.1 Introduction . . . 59
  - 3.1.1 Why a robot? . . . 59

3.1.2 Prior to Dropbot . . . 59
3.1.3 Project publications . . . 60
3.1.4 Chapter objectives . . . 61
3.2 Building the platform . . . 61
3.2.1 Hardware . . . 61
3.2.1.1 Robot frame . . . 62
3.2.1.2 Liquid handling system . . . 64
3.2.1.2.1 Syringe pumps . . . 64
3.2.1.2.2 Mixing stage . . . 64
3.2.1.2.3 Carriage-mounted syringes . . . 66
3.2.1.3 Electronics . . . 67
3.2.1.3.1 Axis control . . . 67
3.2.1.3.2 Carriage control . . . 68
3.2.1.3.3 Syringe pumps control . . . 69
3.2.2 Software . . . 69
3.2.2.1 Robot controller . . . 71
3.2.2.2 Firmware . . . 72
3.2.3 Computer vision . . . 72
3.2.3.1 Image processing . . . 74
3.2.3.2 Blob tracking . . . 77
3.2.3.3 Experimental data generation . . . 77
3.2.3.3.1 Division . . . 77
3.2.3.3.2 Movement . . . 77
3.2.3.3.3 Directionality . . . 78
3.2.4 Artificial intelligence . . . 78
3.2.5 Side development: Microscope . . . 79
3.3 Discussion and future work . . . 80

4 Dropbot results **83**
4.1 Introduction . . . 83
4.1.1 Chapter objectives . . . 85
4.2 Oil droplet system . . . 86
4.3 Robotic set-up . . . 86
4.4 Experimental results . . . 89
4.4.1 Lattice search . . . 89
4.4.2 Baseline experiments . . . 92
4.4.3 Evolutionary experiments . . . 95
4.4.3.1 Using dodecane . . . 95
4.4.3.2 Using octanoic acid . . . 96
4.4.3.3 Using diluted octanoic acid . . . 96
4.4.3.4 Fitness landscapes . . . 101
4.4.3.5 Droplet paths . . . 101
4.4.3.6 Phylogenetic trees . . . 104
4.5 Discussion . . . 107
4.6 Future work . . . 108

5 Flowbot platform **111**
5.1 Introduction . . . 112
5.1.1 While Dropbot was being developed . . . 112
5.1.2 Development from Dropbot . . . 113
5.1.3 Conceptual limitations of liquid handling robots like Dropbot 114
5.1.4 From Dropbot's paper discussion section . . . 114
5.1.5 What could be removed from Dropbot? . . . 115
5.1.6 Embodying evolution . . . 116
5.1.7 Chapter objectives . . . 118
5.2 Designing the device . . . 118
5.2.1 Droplet generation . . . 119
5.2.2 Experimental cycle . . . 121
5.2.3 Oil mixer . . . 122
5.2.4 Four way oil mixer – Final design . . . 124
5.2.5 Future design possibilities . . . 126
5.3 Manufacturing the device . . . 126
5.3.1 Preparing the device to be connected to the tube fittings . . . 127
5.3.2 3D printed examples . . . 127
5.4 Platform set-up . . . 128
5.4.1 Hardware set-up . . . 128
5.4.1.1 From the device to the syringe pumps . . . 129
5.4.1.2 From the syringe pumps to the computer . . . 129
5.4.2 Software set-up . . . 131
5.4.2.1 Genetic Algorithm . . . 134
5.4.2.2 Lattice search . . . 135
5.4.2.3 Image Processing . . . 135
5.4.2.4 Middlware . . . 137
5.4.2.5 Firmware . . . 138
5.5 Generating different environments . . . 140
5.6 Discussion and future work . . . 143

6 Flowbot results **145**
6.1 Introduction . . . 145
6.1.1 Chapter objectives . . . 147
6.2 Oil droplet system . . . 147
6.3 Platform description . . . 149
6.4 Evolutionary experiments results . . . 150
6.4.1 Empty arena . . . 150
6.4.2 Pillars arena . . . 153
6.4.3 Caves arena . . . 157
6.4.4 Hybrid GA runs with environmental changes . . . 157
6.4.5 Experimental reproducibility and control tests . . . 163
6.4.6 Additional statistical data . . . 164
6.5 Lattice search . . . 166
6.6 Relation between droplets and obstacles . . . 166
6.7 Discussion and future work . . . 168

7 Discussion and future work **171**
7.1 Dropbot discussion . . . 172
7.2 Flowbot discussion . . . 173
7.3 Future work . . . 174

8 Experimental **176**
8.1 Dropbot project . . . 176
8.1.1 Robot frame implementation . . . 176
8.1.1.1 Staging area . . . 176
8.1.1.2 X-axis . . . 176
8.1.1.3 Y-axis . . . 176
8.1.1.4 Mobile carriage . . . 176
8.1.1.5 Fluid platform . . . 178
8.1.2 Bill of materials . . . 178
8.1.3 Platform safety . . . 180
8.1.4 Chemistry . . . 180
8.1.4.1 Oils and aqueous phases preparation . . . 182
8.1.4.2 Cleaning cycle . . . 182
8.1.5 Experimental protocol . . . 182
8.2 Flowbot project . . . 183
8.2.1 Platform description . . . 183
8.2.2 Bill of materials . . . 185
8.2.3 Manufacturing the device . . . 186
8.2.3.1 About the filament . . . 186
8.2.3.2 About the 3D printer . . . 187
8.2.3.3 Axon2 software configuration . . . 188
8.2.4 Manufacturing troubleshooting . . . 190
8.2.4.1 Problems while printing . . . 191
8.2.4.2 Problems while testing . . . 193
8.2.5 Droplet generation calibration . . . 194
8.2.6 Lattice search . . . 195
8.2.7 Database . . . 196
8.2.8 Platform safety . . . 196
8.2.9 Protocol . . . 196
8.2.10 Chemistry – preparation of solutions . . . 197
8.2.11 Data analysis – Fitness landscape model . . . 197
8.2.12 Video analysis implementation . . . 198
8.2.13 Middleware implementation . . . 198

Bibliography **203**

1

Introduction

The work that covers this thesis is a multidisciplinary one, going from chemistry to robotic automation:

On one hand, the chemistry we used is related to some origin of life theories, and that is why our introduction will start with a brief review covering from the origin of life to the emergence of protocells (Section 1.1), it will continue with a very basic introduction to surfactant chemistry, which is the main molecule type used in our research (Section 1.2), and it will end up with a review of oil droplets chemistry literature (Section 1.3), being oil droplets a possible particular case of pre-protocellular assemblies.

On the other hand, the concept of life is nowadays very related with the concept of evolution, and this is why a small review of evolving physical entities will be provided here (Section 1.4), in order to consider the question of: "could we do something similar in order to evolve life-like entities?". It will follow with the idea of chemistry automation (Section 1.5), with the objective of reviewing if it would be possible to automate oil droplets based on evolutionary algorithms.

1.1 From the origin of life to protocells

1.1.1 Trying to define *life*

The concept of "life" is one of the most complicated ones that humanity has tried to define for thousands of years. Nowadays, life is defined as a "*characteristic* possessed by physical entities that perform biological processes" ([McIntosh, 2013]). Physical entities are not considered "alive" either because they never were, or because they are deceased. Pairing "life" with "biological processes" can be a loaded statement, because biology itself is the study of life.

In order to be more precise, usually a list of characteristics is given, such as organization, metabolism, homoeostasis, growth, reproduction, response, and evolution ([KhanAcademy, 2016]). A key element in this list with respect to this thesis is the concept of *homoeostasis*, and considering life an intrinsically non-equilibrium process with continuous exchange of matter and energy. The problem with making lists is that it is easy to find counterexamples. Mules, for example, cannot reproduce, but we all would agree that they are alive. Viruses are the other well known example; they are not considered alive, but they would be based on almost any "life" definition. There are also plenty of examples of organisms that can move between the frontier or the inert and the living, being dried seeds the classic example. At

the same time, a wild fire would fulfil all the points of that list[1], except "evolution", although as we will discuss later, "evolution" is not a life requirement for everyone. It would be also possible for a person to build a mechanical machine that would fulfil all those points. Theory says in this case that the persistence of a life entity depends on *form* and not *matter*. Our microscopic structure is constantly changing, and different molecules are flowing in or out. This is not possible with current human-made machines.

The modern concept of evolution completely changed our definition of life. Trifonov [Trifonov, 2011] took 123 definitions of life and calculated that (self-)reproduction and evolution (variation) appeared as a minimal set for a definition: "Life is self-reproduction with variations". Perhaps the first definition of life was given by Democritus (460 BC), who said that the essential characteristic of living things were that they had a soul (*psyche*). Democritus was a "materialist", which means that he thought that everything was formed by matter, including the soul. Aristotle (322 BC) expanded this definition saying that everything in the universe had both matter and form (*hylomorphism*), and the form of a living matter is its soul. Aristotle was also the father of "spontaneous generation" which said that life can appear from inanimate matter in a spontaneous way. Although this theory was disproved by Louis Pasteur when he showed that organisms could not grow in a sterilized environment as opposed to a control environment not sterilized, it is not completely wrong if we consider that planet Earth was at some point inanimate. Finally, a more recent concept is the one of "vitalism", which said that living organisms are different from the non-living ones because they contain a vital principle, similar to the concept of soul. In particular, *vitalism* said that organic matter cannot be synthesized from inorganic components. This hypothesis was discredited in 1828 when Friedrich Wohler synthesized urea from inorganic components. The results from Wohler are not only important because they discredited *vitalism*, but also because they showed a link between chemistry and biology.

We still do not have a conclusive and unambiguous definition for life, or some kind of metric with a set of outputs that would return a binary result, saying if an entity, maybe not even physical, is alive or not. Recent research suggests developing a "Turing test" for life ([Cronin et al., 2006]), or a kind of mathematical formula that would answer the question ([Scharf and Cronin, 2016]). The "Turing test" for life is an interesting one, because as humans we think we are good recognizing life. We usually follow a heuristic approach and focus on basic features we consider to pertain to living organisms, like movement, division, state changes, or the complexity of the physical embodiment.

1.1.2 The origin of life

The Big Bang occurred 13.8 billion years ago (BYA)[2], while planet Earth was formed around 4.5 BYA. Life on Earth is believed to have arisen between 3.5 ([Schopf, 2006])

[1] Fires are considered dissipative structures, because they degrade their environment to maintain their own existence.

[2] The Big Bang is usually considered the start of everything. An interesting question would be what happened before it. Stephen Hawking said that "Since events before the Big Bang have no observational consequences, one may as well cut them out of the theory, and say that time began

and 4.1 BYA ([Bell et al., 2015]). This discrepancy of 600 million years stems from the fact that the earliest ones show fossil evidence of life, while the latest ones show remains of biotic material. This assumes that when Earth was formed it did not contain life. Initially planet Eart was molten, but it cooled down, solidified, and contained water. A very important event, between 4.1 BYA and 3.8 BYA, was the "Late Heavy Bombardment" (LHB). It is still not known if planet Earth contained all the elements necessary to form life or if they came from the outside during the LHB. It is also not known if the LHB quick-started life on planet Earth, although the fact that the LHB era and the fossil record map almost perfectly is very interesting.

The study of the origin of life, or *abiogenesis*, is the study of how non-living matter became alive during early planet Earth. If prebiotic conditions are not considered the study transforms itself from abiogenesis to artificial life. During the rest of this section we will only consider prebiotic conditions, and the study of artificial life will be covered later on during the introduction (Section 1.5.4).

Abiognesis still does not have a clear answer, even though it is studied by many different disciplines, such as biology, chemistry, physics or earth and planetary sciences. Chemistry is the one taking the lead, because life, as we know it, is mostly formed by water and organic molecules, and ultimately any abiogenesis study will need to answer how complex inert chemical systems transformed into simple living ones ([Ruiz-Mirazo et al., 2014]). If we go back to the fossil record and the earliest organisms that we have undisputed proof were on planet Earth, they were already very complex, and the chances of them appearing by some sort of self assembly is not considered valid[3]. The main problem if we consider a full biological pathway for abiogenesis is the concept of *reproduction*. Reproduction is one of the main elements of biological evolution (with variation and selection, see Section 1.4). Reproduction is also an extremely complex process which also had its own evolution. In order to backtrack in time the evolution of reproduction, chemical evolution appears, and also the concepts of *replication* or *repetitive production*. At this point, because there is not reproduction we cannot speak any more about biology, and we need to talk instead about chemical processes. In particular, and always considering prebiotic conditions, chemistry is needed because:

1. The molecules of life, such as lipids, sugars, nucleotides or amino acids, needed to be synthesised. This would include possible precursors.

2. The first functional biopolymers, like polypeptides or polynucleotides (e.g., DNA) should emerge from the polycondensation of the previous molecules of life.

3. The emergence of cellular membranes via for example the self-assembly of surfactant molecules. These boundaries would enclose the elements said before.

4. The ability of some of the biopolymers to replicate themselves, and to couple this replication with the reproduction of said membrane, in order for the system to grow and divide.

at the Big Bang", from `www.hawking.org.uk/the-beginning-of-time.html`.

[3]Going back to the concept of spontaneous generation.

Only after the fourth point it can be said that biology started, thus every step as explained should be performed only using chemistry methods. Figure 1.1 outlines these major steps, from the first step to the left, to the fourth step to the right. At some point during this path life and biology appeared. Because life does not have a definitive definition it is impossible to point exactly where it started. It is important to say that although the ultimate objective of abiogenesis is to provide a full answer, left to right, research nowadays focuses on small portions of this path, starting at any given point[4]. Also, there are in general two different strategies to go through this path. The common one, as the arrow marks, is to go from left to right, which is called *bottom-up*, and aims to increase the complexity of the system in each step. Another strategy goes in the other direction, and it is called *top-down*, and it aims to take working systems and remove elements from it in order to make them simpler for their study. The bottom-up strategy is the common one nowadays. The main problem with top-down is that biology is such a complex system that it is very complicated to remove one of its elements while keeping its functionality and integrity.

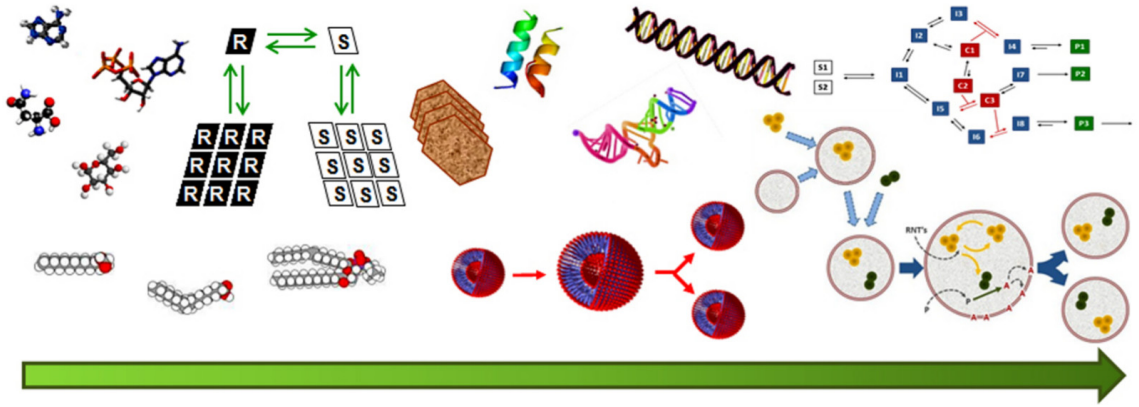


Figure 1.1: Abiognesis scheme, going from the earliest (left) to the latest (right), describing the major steps. It can be said that in the very left everything is chemistry, while at the very end appears life and biology. Reprinted with permission from [Ruiz-Mirazo et al., 2014]. Copyright 2014 American Chemical Society.

Focusing on bottom-up approaches, which are the most extended ones, the different lines of investigation can be divided into at least three different groups.

The first category of experiments tries to find possible geographical locations for abiogenesis on early Earth. They aim for places with a big source of energy and possible prebiotic building blocks. The most known example in this category is the study of deep sea hydrothermal vents. In [Lane and Martin, 2012], for example, the authors initially consider that the oceans were acid. Meanwhile the vents released heat, minerals and alkaline fluid. They claimed that porous structures from the vents separated pockets of warm alkaline fluid from the cold and acidic one from the oceans. The interface between the two created a gradient similar to a battery that fueled the creation of organic molecules.

[4]The results of this thesis, as we will show, use at starting point somewhere near the middle.

A second category of experiments aim to describe how the molecules of life could be synthesised under prebiotic conditions, and also to describe possible pathways to the chemical synthesis of the biopolymers as described before. Within this category, probably the most famous experiment is the Miller-Urey ([Miller, 1953]), where the authors synthesized amino acids from basic precursors like methane, water, ammonia and hydrogen. Another more recent example in the same direction is the work of [Rodriguez-Garcia et al., 2015], which used condensation cycles to form oligopeptides from amino acids. Within this category there is also the famous concept of the "primordial soup", which says that oceans had all the basic elements necessary, and that some source of energy transformed them into organic polymers, and ultimately life. The Miller-Urey experiment can be considered a "primordial soup" experiment, because the authors used a solution with the basic precursors, and a spark as source of energy.

Finally, a third category studies how a general infrastructure could establish itself under prebiotic conditions, and how this infrastructure was implicated in the emergence of living systems. This category focuses on the genetic and metabolic systems, which are considered essentials for life as we know it. Both systems are confronted in the "genes first" vs "metabolism first" schools of thought. This confrontation stems from the fact that life reproduces and evolves using gene - enzyme cycles. The genes explain how to build proteins and enzymes, and these enzymes build more genes. Because this is a closed loop, it creates a "chicken–egg' paradox. Both the genetic and metabolic camps have in common the use of a primordial soup scenario, where all the required elements were available and some source of energy or spontaneous chemical reactions created the more complex molecules required for their respective theories. Finally, the fact that both camps consider different molecules as starting points not only means that they differ in terms of chemistry, but also in a conceptual definition of life. Gene first uses a evolutionary definition of life, while metabolism first uses a thermodynamic one ([Shapiro, 2007]).

The genes first theory considers that initially relatively simple chemical replicators assembled spontaneously in high concentrations, and competed and evolved into more complex species with the ability to replicate, produce proteins, enzymes and some other simpler chemical reactions. The main problem of this theory is that enzymes are considered essential for these processes, thus some basic kind of metabolism was already required. Critics of this theory also claim that the metabolic cycles required for life are extremely complicated, and only possible with the presence of specific proteins. Therefore, they consider that it is very unlikely that such long and complicated sequence of reactions could become organised spontaneously.

The main theory within the genes first camp is the "RNA world hypothesis", which is nowadays under active development and can be considered the mainstream theory of abiogenesis. This hypothesis answers their main theoretical drawback because RNA can act as an enzyme, in a molecule known as a *ribozyme* ([Walter and Engelke, 2002]). This theory starts in a *primordial soup* scenario where nucleotides were available. Nucleotides are molecules that can be considered to have two parts: a base and a backbone. In [Powner et al., 2009] or [Patel et al., 2015] it is shown how nucleotides could form from plausible prebiotic feedstock molecules. These nucleotides would bond with each other through their backbone, forming longer

molecules known as RNA (ribonucleic acid). At the same time, other free nucleotides would be attracted to the bases of the RNA, attaching to it in a process called base pairing, forming a complementary RNA strand. Base pairing can only happen under low temperature conditions, while if the medium is heated, the base pairing would lose its grip, although the strands would be still connected through their backbones. This way, a RNA molecule can replicate itself going through temperature cycles like for example the one explained before in the hydrothermal vents. More interestingly, the base pairing is not always perfect, which means that sometimes the replication can have one or more mutations. From that point on the new RNA strands would compete with each other in order to replicate, in a process discussed briefly before as *chemical evolution*.

The important feature of RNA from a "RNA world" perspective, is that when the molecule is in a low temperature medium, but there are not enough nucleotides available, RNA strands can fold and base-pair with themselves. In this situation, while some of the bases would pair and be unavailable, some of the others are available to the medium, attracting other molecules in their environment in different locations of the strand, and quickstarting chemical reactions. A folded RNA able to carry out a specific reaction is what we called before *ribozyme*. As an example, in [Lau et al., 2004] the authors were able to build a ribozyme that could synthesize nucleotides using molecules available in the environment. Such a ribozyme would potentially be able to replicate faster than other RNA based molecules, winning the chemical evolution. At some points these ribozymes would create enzymes and proteins, and the chemical evolution continued up to the processes known nowadays.

The main problem of the "RNA world" theory is that RNA is already considered a very complex molecule. An example of its complexity is that although we said before that nucleotides could form under prebiotic conditions, it is still not clear how their backbone could form without the use of enzymes. Even binding a base to a polymer backbone is very complicated, and that is why it has also been suggested the possibility of a *proto-RNA* with easier to assemble *proto-RNA bases* ([Cafferty et al., 2013]).

The "metabolism first" camp bases itself on RNA being already a very complicated molecule, and needing instead thermodynamic favourable networks[5] that would perform chemical reactions using small molecules instead. Amino acids, for example, have been synthesized from simpler molecules under prebiotic conditions ([Jiang et al., 2014]), and under similar conditions lipids were also synthesized ([Hargreaves et al., 1977]). Amino acids and lipids could then act as a catalyst for more complex molecules, as in ([Rodriguez-Garcia et al., 2015]) where amino acids formed oligopeptides under condensation cycles. This sequence of reactions would eventually lead to the formation of nucleotides, which were initially used for catalysis and energy storage, and eventually to form RNA.

Something that both camps have in common is the need at some point of a compartmentalisation mechanism. Because life as we know it is embodied into a physical boundary, there is also a school of thought which considers the "lipid world" first ([Segré et al., 2001]), although when compared with the other two main theories, it is very possible that lipids happened in parallel with either metabolism or

[5]Like the discussed hydrothermal vents.

gene systems. In the case of the "metabolism first" theory, a compartment can offer a gradient between the inside and the outside. Also, compartments could contain some degree of internal scaffolding with different conditions, enabling for complex sequences of reactions. In the case of the "genes first" theory, a compartment could protect said ribozymes, while at the same time if such a ribozyme could generate nucleotides, the compartment itself would keep the nucleotides near the RNA and accelerate its replication. This way, a compartment would enable chemical evolution. Compartmentalisation is the central concept of this thesis.

How compartmentalisation could happen will be covered in Section 1.2 from a chemical point of view. The structure responsible of achieving it is believed to be the vesicle ([Chen and Walde, 2010]), which is a supramolecular aggregate with an aqueous interior separated from the aqueous outside through a membrane which is formed by a bilayer of lipids. These lipids could be for example fatty acids, which were a very plausible molecule on prebiotic planet Earth ([Chen and Walde, 2010]). Based on simple chemical conditions and properties, fatty acids can self-assemble into vesicles (Section 1.2).

A final point about the study of abiogenesis is about the possible prebiotic conditions. So far through this introduction we said a few times "under prebiotic conditions", although we never defined them, and this is because the scientific community does not fully agree in a defined set of conditions, and different experiments use different physical and chemical parameters, like temperature, pressure, pH or radiation dose. This does not mean that there is only one answer about the set of conditions, because it is very possible as happens in planet Earth nowadays that different geographical locations had very different conditions where different living organisms could emerge.

If the reader is interested in obtaining more detail about origin of life theories, [Ruiz-Mirazo et al., 2014] offers and excellent and detailed review from a prebiotic chemistry perspective.

1.1.3 Autopoiesis and Chemoton theoretical models

It can be assumed that at some point on planet Earth there were replicating molecules like RNA or DNA, metabolism molecules like proteins, and compartmentalisation molecules like lipids. These three subsystems would work independently or with some basic relation between them, but only their intersection within a unified system can be considered a step forward into abiogenesis. In order to study their interplay, two main theories were postulated which aimed to develop a logical organisation of biological systems. We will start with the concept of autopoiesis from Maturana and Varela ([Luisi, 2003]), and continue with the concept of Chemoton from Tibor Ganti ([Ganti, 2002]).

Autopoiesis is a theory within the field of system biology, which is a field that focuses on the complexity of a whole system considered as a self-organized entity. It focuses on the abstract relations or logical organisation, and forgets about the exact biomechanical processes. Autopoiesis is not about the origin of life, but about its definition. It tries to find a common denominator among all the living organisms in order to separate living from non-living. In particular, it is interested in "life

processes", such as interaction with the environment, evolution and cognition.

Autopoiesis bases its analysis on cellular life as we know it on planet Earth. Cells are very complex systems. However, if we only examine their *basic* functionality as an entity, the system becomes simpler. As [Hanczyc et al., 2008] say: "the complexity of the modern biological machinery obscures the underlying simplicity of the chemical and physical phenomena that gave rise to the first living system". If we consider Figure 1.2, it can be said that a cell as a unit is defined by its boundary, which is a semipermeable membrane that allows the flow in and out of certain substances. The concept of boundary is central in autopoiesis, because the boundary contains a network of reactions which re-generates it and also all the elements required to continue with the chemical network. This way, it is a self-sustainable system. They also define it as an *organisational closure*, meaning that every process or reaction in the system must be possible from other processes or reactions also in the system. This group of process can only work in a collective way, and if one of them fails, the system fails.

Maturana and Varela said: "an autopoietic unit is a system that is capable of self-sustaining owing to an inner network of reactions that re-generate all the system's components". If a bounded system cannot produce its boundary, then it is not considered autopoietic[6]. It is important to say again that the chemical network must built its own embodiment, because this is what differentiates autopoiesis from, for example, auto-catalytic networks in solution. Thus, it can be said that an autopoietic unit is a synonym for living entity: "A system can be said to be living when it is defined by a semipermeable chemical boundary which encompasses a reaction network that is capable of self-maintenance by a process of self-generation of the system's components from within." (From [Luisi, 2003]). If we consider this definition, the system defined at the end of the previous section containing RNA, proteins and lipid boundary would be alive as long as it is embodied, and the RNA and proteins can build both themselves and the lipid boundary.

Something interesting about this theory is that it is not just a *theory* because autopoietic chemical models have been constructed in the lab. They usually focus on a "minimal autopoietic system" which details two competitive reactions: one that builds the boundary, and one that destroys it. If both reactions happen at the same rate, then the system is *homeostatic*. Again, [Luisi, 2003] offers an excellent review of chemical autopoiesis. The criticism such systems usually receive is that in their search of being autopoietic, they simplified the system to a point where it can hardly be said that it represents *life*. From their perspective, as long as the boundary is built by a reaction within it, then it is autopoietic. Most of their research is based on precursors being hydrolysed to form lipid surfactant. From an abiogenesis perspective, the work of [Bissette et al., 2014] is interesting, because instead of using the hydrolysation of a complicated precursor, it used the synthesis of two simpler molecules, that would generate a lipid membrane that bounds the synthesis of these two molecules.

Another similar and contemporary theory is the "Chemoton" as defined by Ganti ([Ganti, 2002])[7]. As with the work of Maturana and Varela, Ganti aimed to define a

[6]Like a virus, which is not considered an autopoietic system.

[7]Although in both Autopoiesis and Chemoton new referenceswere used, these theories go back

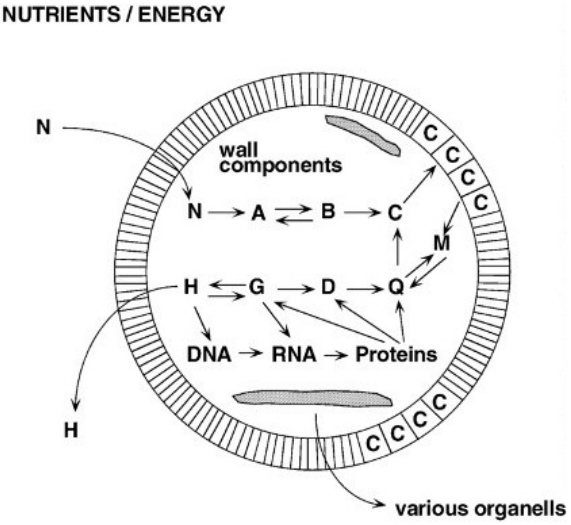


Figure 1.2: Basic outline of a cell's activity. Its membrane is probably the most important concept here, because it is created through an internal network of reactions. The cell can be seen as a machine that maintains itself. This image does not aim to define autopoises. By definiton, for example, an autopoietic system does not include DNA, not because it is not important, but because autopoises only cares about the logic of the system. A cell is an autopoietic system, but an autopoietic system is not defined by a cell. Reprinted with permission from [Luisi, 2003]. Copyright 2003 Springer.

system that would define all the life criteria, and he also focused on the organisational logic of living cells. The main difference between *autopoiesis* and *chemoton*, as we will see shortly, is that autpoiesis focused on a higher level of abstraction, talking about *just* logic processes, while the chemoton took a step forward, and was already speaking about concepts like "metabolism" or "templating".

As said, Ganti based his theory on the cellular organization, and he observed that all the cells function by coupling three distinct subsystems: a metabolism which builds structures using nutrients, a genetic system that controls the metabolism and knows how to build more structures, and a membrane which bounds everything together and allows for nutrients to get in, and waste to go out. With the Chemoton model, he tried to solve this interplay, and explain how metabolism, genes and boundary are *stoichiometrically coupled together*. He defined the objective of chemotons as "multiplying microspheres, controlled by a template process".

Figure 1.3 shows the metabolic map of a chemoton. The membrane allows nutrients X from the outside into the system. An autocatalytic cycle A takes these nutrients and transforms them into membrane precursors T, template monomers V and waste Y. The template monomers accumulate inside the chemoton until a threshold concentration is reached. At this point the template molecule pV_n unzips into two strands, and template monomers *base pair* to them, creating two new template molecules, while also generating waste R. This new waste R combines with the membrane precursors T, creating new boundary molecules which would accumulate

to the 70s, or even earlier.

in the interface, increasing the size of the chemoton, until it eventually divides. This process is not that different from autopoiesis, although there are two major differences. The first one is that the chemoton theory includes the concept of template monomers, similar to, for example, DNA. This way, a chemoton could achieve open-ended evolution, which as we will explain later, many consider the basis of life. The second one is that the chemoton considers the semi-permeability of the membrane a key part of the system, because it would allow nutrients in and waste out. It can be said that this feature allows the chemoton to sense its environment.

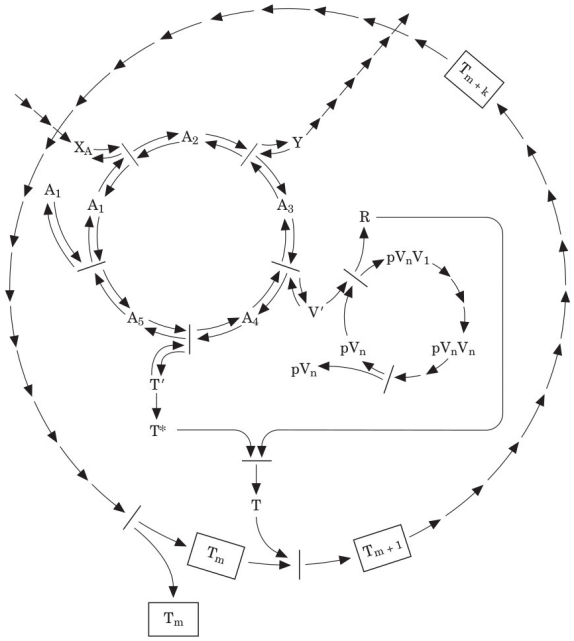


Figure 1.3: Diagram of a chemoton. Metabolism, templating and boundary membrane are are coupled stoichiometrically. Reprinted with permission from [Ganti, 2002]. Copyright 2002 John Wiley and Sons.

1.1.4 Protocells

During the previous sections, when we talk about the definition of life and then about the study of abiogenesis, we avoided the concept of the biological cell in order to tackle those topics from a non-cellular perspective. The biological cell is considered the unit of life, and every living entity known is a cell or is composed by cells. Thus, defining life could also be about defining the biological cell, or studying the origin of life could be about how the first cell formed.

The problem with cells from an abiogenesis perspective is that they are extremely complicated systems, and trying to create one under prebiotic coniditions, or not even prebiotic, is a task no one has achieved yet. If we consider biological evolution and how life on earth seemed to get more complex as time passed, it could be assumed that a simpler similar entity happened before the full-fledged biological cell. This simpler entity is called a *protocell*, and its study focuses on how genes,

metabolism and boundary came altogether into a single unit[8].

It is not enough for a boundary to just *contain* genes and proteins: a protocell must support life, which means that the three fundamental components must be interconnected. The boundary must be semi-permeable, allowing energy and resources to get in, and it also must embody everything together defining a unit. The metabolism uses these resources and energy in order to stay away from thermodynamic equilibrium. It should also construct more boundary, metabolism and information molecules. The information molecules not only control the metabolic cycles, they should also be responsible for storing the protocell's blueprint and transferring it.

This three-way interplay is very similar to what before was described as autopoiesis[9] or chemoton. The main difference between those theories and a protocell is that the protocell should be able to *reproduce*. For a protocell is not enough to increase in size and divide, while capturing metabolic and information molecules inside the new entities. A protocell must perfectly couple its entity division with information template replication ([Szostak et al., 2001]), in a process known as *biological reproduction*[10]. Figure 1.4 shows an example of a possible protocell, with a membrane, DNA and proteins.

The system here described as a protocell is already very complicated. The chances that a compartment, like a vesicle, encapsulated a self-replicating genome and suddenly fully-integrated into a genome-compartment system are very slim. A possible previous step considers a compartment with a partially integrated self-replicating genome, and this concept is known as *protocell precursor* ([Hanczyc et al., 2008]). *Protocell precursors* consider the boundary a passive element which just encapsulated self-replicating genome. A previous step considers that vesicle-like aggregates had to have some sort of *proto-metabolism*. In ([Shirt-Ediss et al., 2015]) the authors considered a situation where vesicles had some inert chemistry inside, and they studied how osmotic flows at different points could act as metabolism, activating different chemical elements. There is even a previous step that just considers far from equilibrium chemistry in lipid compartments, and how these basic "reactors" could model basic protocell features. Although the ultimate objective is to create a full-fledged protocell as described, it can be said that the field of *protocells* considers compartmentalisation a fundamental step, thus it focus greatly on the study of the surrounding membrane[11]. This thesis is an example of work on this direction, and as we will show during our result chapters, we only focused in the membrane.

The study of protocells started more than 100 years ago when scientists tried to synthesize cell-like structures in order to create life. The first research known was

[8]The "RNA world" school of thought might claim that because RNA can represent both genes and metabolism, just RNA inside a boundary can define a protocell.

[9]Autopoiesis is slightly more different because it never talked about DNA or information molecules. Although if we consider that it is a higher level description, those implementation details might not be that important.

[10]We briefly talked before about biological reproduction or chemical replication when talking about why abiogenesis must be a chemical process, and not a biological one. The moment biological reproduction is achieved, it can be considered that biology takes over.

[11]Metabolism (proteins and enzymes) and information templating (DNA and RNA) are covered in great detail in Biology.

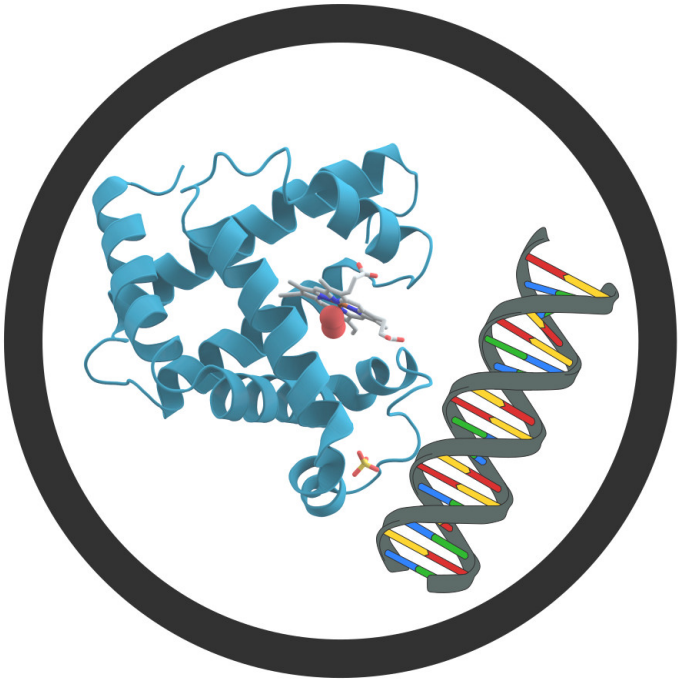


Figure 1.4: Diagram of a basic protocell. A minimal cell should contain genetic material paired with enzymes and proteins inside a boundary. The three systems must interplay together. DNA and protein images from Wikipedia.org under CC BY-SA 3.0 license.

done in 1867 by Moritz Traube who created semi-permeable membranes composed of copper ferrocyanide and a crystal of copper sulfate. This artificial membrane showed selective permeability and responded to osmotic pressure ([Hanczyc, 2008]). Although the best known early experiments are the coacervates from Bungenberg de Jong. Coacervates were a kind of colloid made by mixing and agitating an organic phase, like gelatin, with an aqueous solution. The organic phase would break into small spherical structures which could absorb organic molecules from the medium ([Hanczyc, 2008]). More technically, coacervates are a kind of liquid-liquid phase separation where droplets are held together by hydrophobic forces. Later on, Alexander Oparin in 1965 tried to introduce metabolism into such coacervates. He prepared them containing an enzyme that could polymerize glucose into starch ([Hanczyc, 2008]). Coacervates fell out of fame because they are not considered prebiotically plausible. They are still researched although they are considered nowadays as "artificial cells" or "hybrid protocells' An example of on-going coacervate research is the work of Stephen Mann[12], which we will discuss shortly. Perhaps the most relevant experiment to nowadays research[13] is the one from Irving Langmuir who discovered the interfaces created by fatty acids ([Langmuir, 1917]). Later on, Bangham created osmotically active liposomes (artificial vesicles) using phospholipids which self-assemble into spherical structures similar to cell membranes ([Bangham et al., 1965]).

[12] `http://www.bristolprotolife.co.uk/`
[13] And this thesis.

Nowadays most of the study about protocells follows a *bottom-up* approach. That is, researchers try to build protocells from simpler components, and this approach is what intrinsically we have described here so far. Not only do we want to build a protocell, but we also want to know how it happened. Another less popular approach follows a *top down* path. It tries to *reverse engineer* the cell. They usually take existing cells and try to remove elements in order to simplify their complexity. There are two problems associated with this approach. The first one is that current cells can barely tolerate any kind of simplification. The second one is that this approach is not really answering how life emerged. Therefore, from an abiogenesis perspective, it is not that useful and rarely used. An example of *top down* approach is the work in [Glass et al., 2006] where the authors managed to reduce the genetic set from *mycoplasma genitalium*, which has the smallest genome of any organism known, from 482 to 382 genes.

Within the *bottom up* approach, most of the research focuses on how simple prebiotic molecules could self-organise into more complex structures, very similar to what we have explained so far. This research usually focuses on simple membrane effects, like how for example could protocells divide and move (see Section 1.3). Another *bottom up* approach tries instead to incorporate into protocells existing functionality from biological cells. An example of research in this direction can be seen on [Krishna Kumar et al., 2011], where a structure mimicking a cytoskeleton made using polymer gel was encapsulated into a vesicle. Another example is the work shown in [Ichihashi et al., 2013] where artificial RNA was introduced into water-in-oil droplets and replicated for more than 600 generations. They objective was to show that RNA can replicate inside an artificial cell. Although this kind of research steers away from pure prebiotic conditions, it allows for the research of situations that were very likely to happen, but that we do not have still the technology to arrive to that stage using only prebiotic conditions.

Another topic of discussion is about whether the first protocells used organic or inorganic molecules. It is universally agreed nowadays that organic matter was present during prebiotic planet Earth. Hydrogen cyanide, for example, has been found in comets and other space bodies ([Matthews and Minard, 2006]), thus it could be very well be present on planet Earth from the first moment, or it could arrive during the LHB. Not only that, but more complex organic molecules than HCN have been found in outer bodies, like amino acids ([Altwegg et al., 2016]), or even fatty acid vesicles ([Chen and Walde, 2010]). Nevertheless, inorganic protocells are studied, and an example of this is also the work of Stephen Mann who used silica nanoparticles as boundary in water-in-oil biphasic systems. Their membrane was semi-permeable and they could encapsulate inside cell-free gene expression and enzyme catalytic reactions ([Li et al., 2011]), thus this work would also fall into *hybrid protocells*. Another example of inorganic protocells is the work described in ([Cooper et al., 2011]), where an inorganic macromolecule (polyoxometalate or POM) was used as building block to synthesize the membrane. The POM, containing small cations, was extruded using a nozzle into a bulk solution containing large anions. The exchange of ions would create an insoluble aggregate which formed a membrane. This membrane was selectively permeable, it could be deflated and inflated several times, and it could host new smaller membranes inside.

Perhaps the most famous research in the field of protocells is the work done by Jack Szostak, who in 2001 ([Szostak et al., 2001]) already talked about concept of a vesicle containing a fully integrated RNA-ribozyme that played a central role in the unit survival, not only from a purely metabolism point of view, but also competitive-natural selection one. He defined a simple example as a ribozyme that would synthesize amphipathic lipids enabling the membrane to grow[14].

Szostak's research focuses for example on growth and division of vesicles without loss of internal content ([Zhu et al., 2009]), where the authors encapsulated RNA inside the vesicles to show that the division was *clean* although the RNA did not do anything. In order to achieve the growth-division cycles they used plausible prebiotic conditions, which he claimed were similar to evaporation and rain ([Szostak, 2016]). In another research ([Adamala and Szostak, 2013]) the authors also showed how DNA can replicate inside fatty acid vesicles by adding nucleotides to the outside of the vesicle. This research shows how membranes could encapsulate information molecules. He also explored the concept of competition, which is critical for natural selection and evolution. In [Budin and Szostak, 2011] he created populations with different ratios of phopholipids, and he showed that the vesicles with more phopholipids grew at the expense of the others. Although perhaps the research that has been nearer of achieving what Szostak described is [Kurihara et al., 2011], where the authors managed to couple the self-replication of DNA with the self-reproduction of a lipid vesicle. More importantly, they managed to transfer the new DNA into the daughter vesicles. Their main limitation from a prebiotic point of view is that they used polymerase chain reaction (PCR) to amplify the DNA.

Finally, although in this thesis protocells are covered from an origin of life perspective, artificial cells can have some other applications. For example in [Lentini et al., 2014] artificial cells were used in conjunction with *E. coli* in order to translate a chemical message that *E. coli* could not sense on its own synthesizing a molecule that would activate a response from the natural cell. Another example is the work of [Gardner et al., 2009] where the authors created an artificial cell that could synthesize sugar, and this activated a bioluminescent response in neighbour bacteria. A last example is some of the work done in the Evobliss project[15], where the authors aim to use artificial cells to clean the biofilm generated in microbial fuel cells.

1.2 Surfactant chemistry primer

1.2.1 Chemical polarity

The nucleus of an atom is formed of two different subatomic particles: protons, which are positively charged, and neutrons, which are electrically neutral. Surrounding the nucleus there is a "cloud" containing electrons, which are negatively charged and

[14]It seems that recently Szostak is steering away from the "RNA world" theory, but whether it is a ribozyme or some other gene-metabolism mechanism, it does not really matter.

[15]The Evobliss project is an european one, and the Cronin group is part of it. In particular, the Chemobot team, which is the team that represents the work of this thesis. `https://blogit.itu.dk/evoblissproject/`

occupy most of the size of the atom[16]. If the number of electrons and protons is the same, the atom is electrically neutral. The number of electrons and protons might differ, because the number of electrons can vary. Depending on the difference between electrons and protons, an ion can be positive or negative, in which cases ions might bond to each other in order to form electrically neutral chemical compounds.

There are three major types of chemical bonding interactions:

1. **Ionic**. It happens when one electron is directly transferred from one atom to another. This can occur when the receiving atoms allows the transferred electron to be nearer its nucleus than the original atom. Once the bond materializes, the atom that lost the electron is positively charged (anion), and the one that received it is negatively charged (cation). An example of these compounds are salts, like sodium chloride (NaCl).

2. **Metallic**. Metal atoms can arrange into a regular pattern called a *lattice*, and share a *sea* of electrons where each individual atom donates one or more electrons. These electrons can be associated with many different atoms at once. An example of these compounds are gold, silver or copper. The *sea* of electrons is what makes these compounds conductive.

3. **Covalent**. If we consider what was explained before about an atom being formed by a nucleus and a cloud of electrons, in a covalent bond the atoms share this electron cloud in order to form a molecule. Examples of covalent bonds forming molecules are oxygen (O_2), water (H_2O) or methane (CH_4).

Because in a covalent bond the electrons are *shared* between atoms, it is important to know how the new cloud of electrons will be distributed among them. In particular, there is a property called *electronegativity* which explains the ability of an atom to attract bonded electrons within a molecule. We can assume that if a molecule only contains one type of atom, like H_2, the nuclei share equally the cloud of electrons. On the other hand, in a molecule with different atoms the bond is said to be polar if the atoms have different electronegativities, because the cloud of electrons will not be shared equally. Water, for example, contains two atoms of hydrogen attached to one of oxygen. The electronegativity of hydrogen is 2.2, while the electronegativity of oxygen is 3.44. Therefore, in a water molecule the bonds between hydrogen and oxygen are polar, with more of the electron cloud being allocated to the oxygen than the two hydrogen atoms. The fact that a molecule contains polar bonds does not necessarily makes it a polar substance. Ultimately it will depend on its three-dimensional shape. Figure 1.5 shows different polar and non-polar molecules. In the case of carbon dioxide and methane, even though they contain polar bonds, its 3D structure is symmetrical which creates a uniform electron distribution, thus they are not polar molecules. In the case of water, because the angle between the hydrogen atoms and the oxygen has 104^o the electrons are unequally shared, making it a polar molecule.

Figure 1.6 left, shows the electrostatic potential map of a water molecule. Although the molecule is neutral, the electrons in the O—H bond are unequally shared,

[16]The nucleus occupies about 1/10 000 the size of the atom

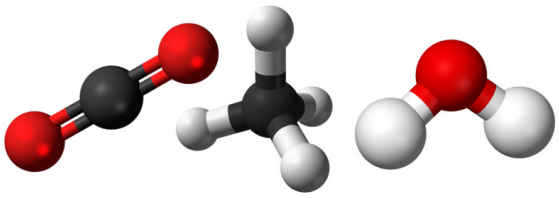


Figure 1.5: Polar and non-polar molecules. From left to right: CO_2 (non-polar), CH_4 (non-polar), and H_2O (polar). Red atoms are oxygen, white atoms are hydrogen, black atoms are cabron. Images from Wikipedia.org released under public domain.

because the oxygen atom pulls the shared electrons. This means that there is a net negative charge on the oxygen atom, while there is a net positive charge on the H atoms. This way, a water molecule is like a magnet, with a positive end and a negative one. In chemistry, instead of magnet it is called *dipole*.

When several water molecules are put together in a container, they move around and collide with each other. As they move past each other, they align themselves so that the positive end of one molecule is oriented towards the negative end of another molecule, see Figure 1.6 right. This behaviour between polar molecules is called dipole-dipole force, and it is present in any substance formed by dipole molecules. It is important to note that the molecules would continue to move and change neighbours, but they would always maintain this dipole-dipole configuration.

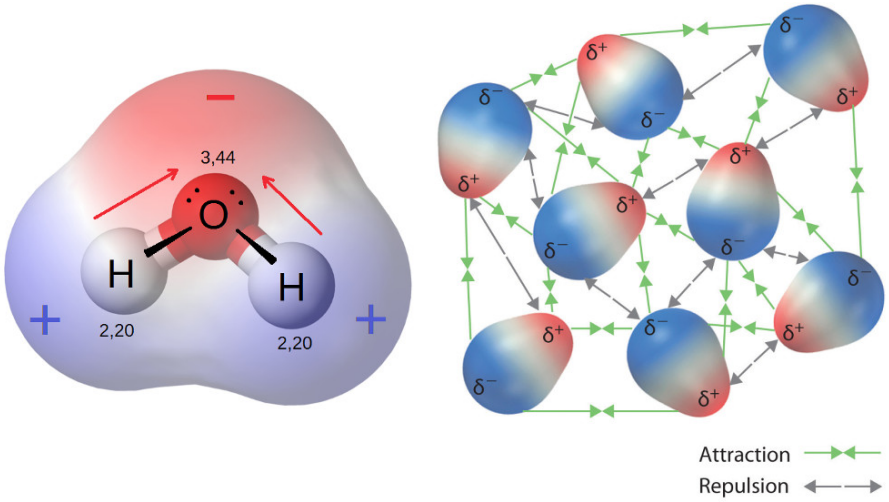


Figure 1.6: Left: Water's electrostatic potential map. **Right:** Dipole-dipole forces. In the case of water, hydrogen bonds are formed between hydrogen atoms from a molecule, and oxygen atoms from another one. Images from Wikipedia.org released under public domain.

In the particular case of water, and some other substances containing bonds between hydrogen and either oxygen, nitrogen or fluorine, a strong case of intermolecular dipole-dipole interaction happens, called a *hydrogen bond*, between an H atom from a molecule, and an O, N or F atom from nearby molecules. The water molecules in Figure 1.6 are bonded through hydrogen bonding. This "strong

intermolecular force" happens because the electronegativities of oxygen, nitrogen or fluorine are the highest ones, and when they bond with hydrogen they create a very polar bond, pulling almost completely hydrogen's only electron, exposing its positive nucleus. Although the hydrogen bond is weaker than a covalent bond, it is the strongest intermolecular force. The hydrogen bond can also happen within a molecule, and it is common in DNA, protein folding, and even synthetic polymers like nylon. All the other dipole molecules not containing O, N or F atoms will form dipole-dipole force interactions, which are considered weaker than a hydrogen bond.

Non-polar substances don't have a dipole, but they must have some other force of attraction, otherwise they would only exist in gas phase. Their force of attraction is called *disperson forces* or *van der Waals forces*. If we consider a big quantity of non-polar substances together, even though the electrons are symmetrically distributed within each molecule, the global cloud of electrons might not symmetrical. This is called *instantaneous dipole*. If the cloud of electrons was measured and averaged over a period of time it would be symmetrical, but if we take an instant snapshot, it might not be.

Dispersion forces are weaker than dipole-dipole forces, which at the same time are weaker than hydrogen bonds. If a polar and a non-polar substance were mixed together, two separated phases would appear, mostly because the polar molecules would cluster together, leaving the non-polar molecules aside. It can also be said that polar and non-polar fluids are immiscible. The interface between the two immiscible phases has a high inter-facial tension. For example, the inter-facial tension between nitrobenzene (non-polar) and water is 27 mN/m ([Hanczyc, 2014]).

1.2.2 Amphiphiles

There are, however, certain molecules that contain a polar part and a non-polar one. These molecules are called *amphiphiles*. Figure 1.7 shows two of them. The top one is called octanoic acid. This molecule is a fatty acid, which is a class of molecules we discussed previously as being plausible in prebiotic conditions. It is also one of the main molecules used during this research (see Section 8.1.4, p.180). The bottom one is called 1-palmitoyl-2-oleoyl-sn-glycero-3-phosphocholine (POPC), which is a phospholipid[17] like the ones that form natural cell membranes. Octanoic acid, as any fatty acid, contains a carboxylic acid as head, and a aliphatic chain. The carboxylic acid is very polar, while the carbon chain is not. Considering how the molecule interacts with water, this means that the head is hydrophilic, while the tail is hydrophobic. There are many more molecules which are amphiphiles. Fatty alcohols, which were also used during this research, are for example also amphiphiles, or phospholipids, which are the molecules that form the lipid bilayers of all cell membranes.

Molecules that contain a hydrophobic group and a hydrophilic one are also called *surfactants*. Surfactant means *surface active agent*, and they are called this way because they adsorb at interfaces, whether it is water-air or water-oil, with the objective of reducing their inter-facial tension. Their polar head will diffuse in the

[17]Phospholipids are called this way because they have a phosphorus based "head" and at least one lipid tail.

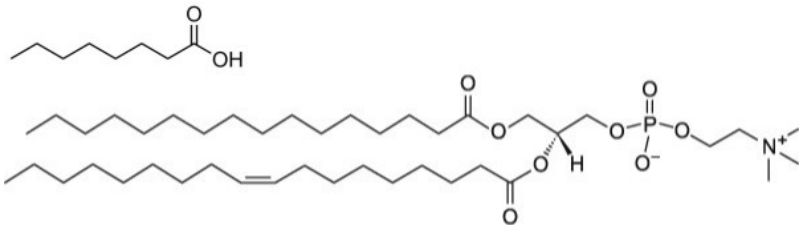


Figure 1.7: Top: Octanoic acid molecule. It is an amphiphile because it contains a polar head (carboxylic acid) and a non-polar tail (aliphatic chain). **Bottom:** 1-palmitoyl-2-oleoyl-sn-glycero-3-phosphocholine (POPC), a common cellular lipid. It contains a polar head and two non-polar carbon chains. Images from Wikipedia.org; released under public domain.

water phase, while the non-polar tail will extend to the oil or air phase, see Figure 1.9 top left. The difference between *amphiphile* and *surfactant* is not clear, and often they are used synonymously in the literature. Amphiphile refers to the nature of the molecule, while surfactant refers to its condition when it acts at the interface of two unlike materials. Finally, surfactant molecules are very common in our daily life. Detergent, for example, is a surfactant which attaches its tail to oil / fat groups, while attaching its head to running water, this way it becomes easier to clean the oil / fat away.

1.2.3 Surface tension

Water molecules are dipoles, therefore they created a high force of attraction between them. Water molecules in the bulk liquid are equally attracted in every direction by other water molecules, but the ones in the surface are not because some of their neighbour molecules, like oxygen or hydrogen in the case of a water-air interface, have a lower polarity, which means that water molecules in the surface are strongly pulled inwards, creating a high surface tension.

When a substance with a lower surface tension, like a surfactant, is added to water, a gradient is created because the surface tension across the whole surface is not equally distributed. This generates a surface flow from low to high, because the liquid with a higher surface tension pulls stronger, until the global surface tension is again equally distributed. The mass transfer caused by the surface tension gradient is called Marangoni flow ([Scriven and Sternling, 1960]).

1.2.4 Surfactant Classification

Surfactants are classified based on the charge of its polar head group. They are classified as anionic, cationic, non-ionic and zwitterionic ([Holmberg et al., 2002]). Zwitterionic surfactants contain both an anionic and cationic charge under normal conditions. There are also amphoteric surfactants which based on the pH can be either cationic, anionic or zwitterionic. Most ionic surfactants are monovalent, but there are also divalent examples. The choice of the counterion plays a role in the physiochemical properties. For a few surfactants there is ambiguity on their classification because their charge may be pH dependant or depend on other physical

properties.

The main difference between ionic surfactants and non-ionic surfactants is that ionic surfactants produce charged surfaces, while non-ionic surfactants produce a protective coating.

Anionics surfactants are those whose polar head is negatively charged, usually carboxylate, sulfate, sulfonate or phosphate groups. Approximately 60% of the worldwide surfactant production are anionic. The counterions most commonly used are sodium and potassium for water solubility, ammonium and calcium for oil solubility and various alkyl amines for both water and oil solubility. Soap is the most used type of surfactant, produced from animals fats or vegetable oils. Generally anionic surfactants are not compatible with cationic ones.

Cationic surfactants are those with a positively charged polar head, usually a nitrogen atom, commonly with amine or quaternary ammonium-based products. Amines cannot be used with high pH, while quaternary ammonium compounds are not pH sensitive. The majority of surfaces, metals, minerals, plastics, fibres and cell membranes are negatively charged, therefore cationic surfactants bind to these surfaces, giving special characteristics to them. For example as anticorrosion agents, dispersants, hair conditioners, anticaking agents or bactericides. They are the third largest class of surfactants, and they generally are not compatible with anionic surfactants.

Non-ionic surfactants usually have a polyether or a polyhydroxyl unit as the polar group. They are commonly used in liquid and powder detergents as well as a variety of industrial applications. They are particularly useful in stabilizing oil in water emulsions. They are the second largest class or surfactant, they are normally compatible with all other types or surfactants, they are not sensitive to hard water and they are usually temperature dependent.

Zwitterionic surfactants contain two charged groups of different polarity. Usually the positive charge is ammonium while the negative one is usually carboxylate. They may be confused with amphoretic surfactants. An amphoretic surfactant is one that goes from cationic to anionic, via zwitterionics, based on the pH. Usually the compound is only zwitterionic over a certain pH range. Zwitterionic surfactants are characterized by having excellent dermatological properties because they show very low eye and skin irritation, which makes them very good for personal care products. They are the smallest class because they are very expensive. They are compatible with all other surfactants. They are not sensitive to hard water, and they are generally stable in acids and bases.

1.2.5 Aggregates

When a single surfactant molecule is added into a polar solution, like water, its head will remain in the water phase, but the tail will extend to the air phase, which means that the molecule will adsorb into the water-air interface, see Figure 1.9 top left image. If water is mixed with a non-polar substance, they will form two separated layers, and the surfactant molecules will again adsorb to the interface, with the head inside the water phase, and the tail extending to the oil phase. Only surfactant unimers contribute to lowering surface and interface tension between two

different phases.

If surfactant molecules are continually added into the water phase, they will continue to go to the water-air or water-oil interface, until at some point the surfactant concentration will arrive to what is known as *critical micelle concentration* (CMC) or *critical vesicle concentracion* (CVC). From this point on, new surfactant molecules will aggregate into micelles or vesicles depending on the surfactant characteristics. In particular, it depends on its *packing parameter*, which relates the size of its head, to the length of the tail and the total volume of the molecule ([Israelachvili et al., 1977]), see Figure 1.8. The packing parameter can also depend on characteristics of the medium, like its pH. Surfactants can aggregate into more different structures, but micelles and vesicles are the most common ones.

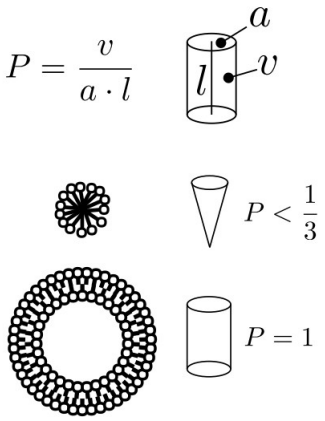


Figure 1.8: Surfactant packing parameter which determines if it will form vesicles or micelles. There are many more aggregates it could form, but these two are the most common ones. This image is from [Shirt-Ediss, 2016], with license CC-BY-SA 4.0.

If the surfactant tends to form micelles, depending on the bulk liquid, surfactants will aggregate into reverse micelles (Figure 1.9 A) when the bulk liquid is oil, or micelles (Figure 1.9 B) when the bulk liquid is water. Micelles are aggregates where the tails are joined together into a non-polar compartment while the heads are exposed to the water phase. Reverse micelles follow the same strategy, but creating a polar compartment instead.

If the surfactant tends to form vesicles, once the concentration arrives to its CVC, new surfactant molecules will join their tails together forming a lipid bilayer (Figure 1.9 E) which will create a water compartment in a water medium. They can also form reverse vesicles when the medium is oil, but they are very rare. Vesicles forming naturally tend to contain nested smaller vesicles inside, and multiple bilayers stacked to each other. These vesicles are called "multilamellar"[18]. Usually when created in the lab, unilamellar vesicles are preferred because they are easier to study.

Finally, surfactants can also generate droplets, like in Figure 1.9 C, where there is a water-in-oil droplet, or in Figure 1.9 D where there is an oil-in-water droplet. They are formed when a droplet-sized volume of liquid is transferred into another

[18]This is very interesting from a proto-metabolism perspective, as explained before.

one where it is insoluble. Without surfactant molecules, the transferred liquid might form droplets initially, but it will eventually disperse into a layer. With surfactant molecules, it will form droplets, avoid their coalescence and help with their stability. Although droplets can be fully made with surfactants, they will usually contain a bulk phase of water or oil. This thesis focuses on oil-in-water droplets.

Vesicles, micelles and droplets are very dynamic structures, and the molecules that compose them are continuously being exchanged with free unimers from the medium. In the case of vesicles, there is also a flip-flop dynamic, where a molecule forming part of the membrane would either flip or flop continuously to be either in the inner or outer layer.

It is important to note that all these supramolecular aggregates self-assemble through the action of weak forces, like the ones explained before. Depending on the exact atomic composition of the surfactant molecule, it could be either via the use of hydrogen bonds or using *van der Waals* forces. Also, all these aggregates structures can contain different types of surfactant molecules.

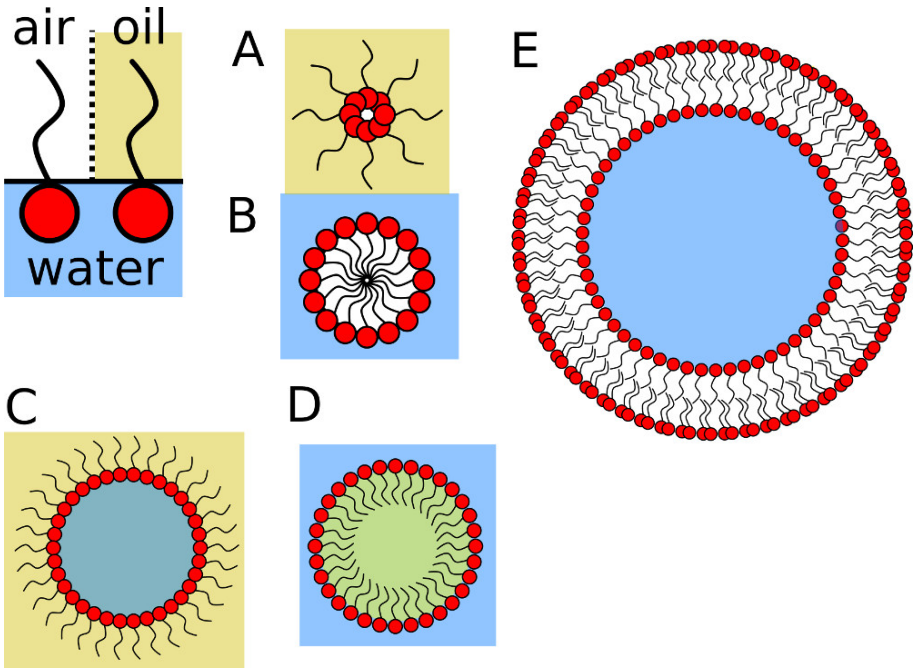


Figure 1.9: Surfactant aggregates. **Top left**: Unimers adsorb to the water-air or water-oil interface. When their concentration surpasses a threshold, they will either form **(A)** reverse micelles, **(B)** micelles or **(E)** vesicles, depending on the packing parameter of the molecule. Surfactant molecules with a bulk phase can also form **(C)** water-in-oil droplets or **(D)** oil-in-water droplets.

1.2.6 Surfactants and the origin of life

As just explained, fatty acids can easily self-assemble into macromolecular aggregations (Section 1.2.5). The main three structures we discussed about: micelles, vesicles and droplets; are very interesting from an origin of life perspective because

they create compartments similar to natural cells, and in particular vesicles are strikingly similar with their lipid bilayer, and their ability to create water compartments in a water medium. Also, as discussed before, there are many reports of fatty acids or similar molecules being found in outer space bodies, like for example in [Yuen and Kvenvolden, 1973]. There are also many reports about synthesizing them from basic components under prebiotic conditions, like for example in [Dworkin et al., 2001]. Therefore, we know that fatty acids and other similar surfactant molecules were available during prebiotic Earth, and we also know that these molecules easily aggregate into compartments. Thus, we know with certainty that such compartments were available during prebiotic conditions. Moreover, we showed during Section 1.1.4 that current research has used fatty vesicles as protocell models in which to perform DNA/RNA replication, enzymatic reactions and many other life-like processes. The question we do not have an answer yet for is if they had a role during the origin of life or if they were mere bystanders.

What we know for sure is that natural cells use phospholipids to build their membranes. Although meteorites can contain phosphorus, prebiotic syntheses of modern phospholipids has returned very low yields, which minimizes their possible role during early compartmentalization ([Hanczyc and Monnard, 2016]). On the other hand, using the concept of *chemical evolution* it has been reported how gradual alterations of single chain amphiphiles could have led to the formation of phospholipids ([Monnard and Deamer, 2002]). Therefore, it is possible that current phospholipid membranes *evolved* from fatty acid membranes.

There are three good features regarding fatty acids membranes when compared to phopholipid membranes from a prebiotic point of view ([Hanczyc et al., 2008]): (1) Fatty acids form very strong membranes as measured by the maximum membrane tension (t*). While phospholipid membranes have a t* between 3 and 40 mN/m, oleic acid vesicles have a t* of 10 mN/m, for example, which falls in the range of phospholipid membranes. This indicates that they have a similar tolerance to osmosis. (2) Their dynamic behaviour (growth, membrane transfer, pH gradients, ...) is very fast when compared to phospholipids membranes by several orders of magnitude. This would accelerate possible chemical evolution. (3) Fatty acid vesicles are permeable to substances important for enzymatic activity, while phospholipid membranes are not. Current natural cells use proteins channels in order to perform this transfer mechanism, but the production of these channels already requires high complexity in terms of metabolism.

Some of the problems associated with fatty acid vesicles from an abiotic perspective is that they are only stable under a limited range of conditions. One of their main limitations is pH, because their self-assembly processes depend on the formation of *hydrogen bonds*. Salt concentration in the medium can also be a factor, and oceans contain a lot of salt, which would lead to extreme osmotic effects up to the point where a vesicle would burst. On the other hand, it is considered that mixing fatty acids with similar hydrocarbon chain lengths molecules can lead to a wider range of conditions ([Hanczyc and Monnard, 2016]), and considering that prebiotic Earth was a big *mess* the possibility of membranes with mixed molecules is very high.

1.3 From protocells to oil droplets

During this introduction we have tried to focus as much as possible on plausible prebiotic structures and processes. The central element so far has been the protocell, which is considered among most researchers a stepping-stone in abiogenesis, because it is the first system that coupled information, metabolism and membrane molecules. Not only these molecules had to exist together in a single entity, but their creation and maintenance had to be related and interconnected. In particular, we focused on the membrane, and why vesicles are considered the best candidates (Section 1.2.6), because they self-assemble from plausible prebiotic components, and because they offer a water compartment in a water medium, and as far as we know water is requirement for life. The odds of information and metabolism molecules simply going inside a vesicle and self-replicating are non-existent. Thus, vesicles cannot be considered passive actors during abiogenesis. Moreover, it is believed that as happened with information and metabolism molecules, membrane molecules also went through chemical evolution, to the point that they were already advanced compartments when all three subsystems were put together in order to form a protocell.

Ben Shirt-Ediss described in his PhD thesis ([Shirt-Ediss, 2016]) an excellent reformulation of the major prebiotic transitions from a compartmentalization point of view, which we will describe[19] here:

1. Supramolecular aggregates formed from a mixture of lipids and other hydrophobic molecules. They were not hosting chemical reactions. Vesicles were passive objects, and they competed for lipids available in the medium. Because every structure had a different composition, they had different success ratios across different environments where lipid monomers were scarce. It is not clear if at this point we can speak about chemical evolution or purely stochastic processes. As we will discuss in Section 1.4, some researchers argue that these aggregates already contained *compositional information*.

2. Vesicles self-assembled from available lipids or other similar molecules in the external medium, encapsulating basic chemical reactions inside their aqueous interior or inside their membrane. Their complexity was similar to the medium, but far-from-equilibirum dynamics were present, like chemotaxis.

3. Vesicles encapsulated chemical reactions that created new membrane molecules not present in the external medium. Their aqueous interior started to be more complex than the medium. This marks an important step from self-assembly to self-reproduction. This can be considered the start of chemical evolution from a compartmentalization perspective.

4. Vesicles encapsulated more complex reaction networks, generating species like lipids, peptides or catalysts. These vesicles were able to control their membrane composition from the inside, giving them a bigger chance of surviv-

[19] It is not a verbatim copy, but an adaptation to fit this thesis. Any mistake is solely our responsibility, and is unrelated to the original author.

ability in different environmental conditions. They could manifest non-linear behaviours.

5. Vesicles could generate template biomolecules through their encapsulated metabolism. These vesicles would be able to divide and transfer their information to their offspring. We can consider this a protocell, and perhaps the start of biological evolution.

6. Full-fledged biological cells and open-ended evolution.

It is important to note that in the first four steps the entities were not considered protocells yet, and that step 3 is the latest one with experimental results, like all the chemical autopoiesis experiments described in [Luisi, 2003].

Through all these prebiotic transitions, right from the start, vesicles are the common denominator. Our research from now on will focus on droplet emulsions instead, which are not considered plausible abiogenesis structures that could end up forming a cell. Their main drawback is that droplets offer an oil-in-water or a water-in-oil compartment, while life requires a water compartment in a water medium. Also, if there were droplets, there had to be vesicles, because both structures self-assemble under similar conditions. Thus, we can also reject the idea of droplets being vesicle precursors.

The reason for choosing droplets is purely technical: they are easier to generate. This has the advantage of enabling the exploration of abiogenesis scenarios which might be not as easy to explore with other structures, like vesicles, because the process of generating them is already complicated enough to limit the possible applications. It can be said that droplets offer an easier experimental framework, but in order to get it we had to steer a bit away from plausible abiogenesis structures. Droplets did exist under prebiotic conditions, although they were probably unrelated to the origin of life. Hence, even though droplets might not be an abiogenesis candidate, the processes and behaviours they are capable of can be life-like, allowing us to study how such phenomena could happen under *almost* prebiotic conditions.

Another positive side-effect of choosing droplets is that because they are not possible protocell precursors, it increases the range of conditions we can work with, because we are not tied to prebiotic ones any more. This means that not only can we explore the emergence of "natural cells", but we can also explore the concept of "artificial cells". Although in order to not go completely out of topic we will only consider artificial cells that can exist with partial prebiotic conditions (more on this in Section 1.5.4). For example, we will not consider the case of mechanical cells, or electronic ones, or any other type of cell using modern technology. During the rest of this section, we will focus on droplet macroscopic behaviours that could be linked to abiogenesis, either to explain natural processes that could occur in natural cells, or to build abiogenesis-based artificial cells.

Going back to the major prebiotic transition steps, we will focus on the first two. We will consider droplets that aggregate from lipids available in the system, that can have far-from-equilibrium dynamics, and that can host chemical reactions, either in the compartment, in the membrane, or in the medium.

Far-from-equilibrium systems are perhaps the most interesting ones from an *origin of life* perspective, because as we explained during Section 1.1.4, basic physical effects like osmosis are considered very related to some of the initial proto-metabolism in vesicles (e.g. [Shirt-Ediss et al., 2015]). These systems are easy to replicate in oil and water emulsions, because the oil-water interface acts as a semipermeable barrier between water and oil where surfactant molecules flow in either direction. These flows create inhomogeneous instabilities across the interface which can deform it, push it, pull it, or many other different manifestations ([Sumino et al., 2009]) known as Maragoni effects (discussed in Section 1.2.3 and [Scriven and Sternling, 1960]). This effect is achieved by starting an experiment from a non-equilibrium state, where both the oil and aqueous phase contain opposite elements of the same property, generating a flow through the interface until the system arrives to an equilibrium state. Surfactants are the key molecule to this kind of emulsions, and, for example, by having differently charged surfactants in each phase, flows through the interface will be generated until the charges of both phases neutralise.

In such systems, droplets become non-equilibrium entities, and can display behaviours that cannot be observed at equilibrium. Our focus is on behaviours similar to the ones displayed by living cells, like movement, division, communication with the outside and other cells, or shape changes. The question is then, if oil in water emulsions can emulate some of those life-like behaviours, and if they do, if a link to abiogenesis exists. The following sections will contain a brief literature review about droplet chemistry focusing on this kind of behaviours. We will start with movement and division, which are the most researched life-like behaviours. We will follow with "multicellular" behaviours between droplets, and then some other life-like behaviours. The last section will be about interesting behaviours or droplet applications which are not related to "life", but which are *state of the art* droplet research.

Possibly the field with more droplet-based research is the one using microfluidic devices, but we will focus instead in macro-droplets of at least around 1 µl of volume[20], like the ones a researcher could generate manually using a standard pipette. Our reasoning for focusing on these bigger droplets is that those are the ones that we will use in our platforms, as we will explain later in this thesis.

1.3.1 Movement

The most famous work in terms of self-propelled droplets is the research from the Sugawara group ([Hanczyc et al., 2007] and [Toyota et al., 2009]). We will concentrate on their first publication from 2007, because the authors used fatty acid molecules similar to ones described before (Section 1.2.2). Their droplets contained nitrobenzene as the bulk phase, and oleic anydride as "fuel", while the aqueous phase contained oleic acid in water (pH 11). Their aqueous phase had a volume of 100 µl while their droplets had a volume of 0.2 µl, although in subsequent publications they have reported that the system works in bigger volumes, like 1 ml for the aqueous phase and 5 µl for the aqueous phase. Once a droplet was placed in the surfactant medium, oleic acid bound to the oil phase, making the droplet stable, and

[20]Microfluidic devices work with droplet volumes from the pico to nano level.

initially no movement was observed (Figure 1.10, Stage 1). Eventually the coating surfactant would break the symmetry of the droplet surface, and internal structures appeared in the oil phase, moving in oscillations and exposing oleic anhydride to the alkaline media, enabling its hydrolysis into oleate. Red circles in Figure 1.10 "Stage 2" mark possible hydrolysis points. This had a localised effect over the global surface tension of the droplet. The flows caused by these surface tension instabilities pushed the oleate surfactant to the rear part of the droplet (Figure 1.10, Stage 3, blue arrows). At the same time, the hydrolysis reactions concentrated in the front of the droplet. This created a pH gradient through the whole droplet that generated two counter-rotating vortices within it, making it move forward. Depending on the quantity of fuel the droplets could move for hours. Their follow-up research, from 2009, was similar, but in this case the fuel was supplied from the aqueous phase while the oil droplets contained a catalyst that used the external fuel source to power its movement. This system had the advantage that the external fuel could be added dynamically to the aqueous phase, enabling the droplets to move indefinitely. In both cases the droplets had an apparent stochastic pattern of movement, which they analysed in a subsequent publication ([Horibe et al., 2011]).

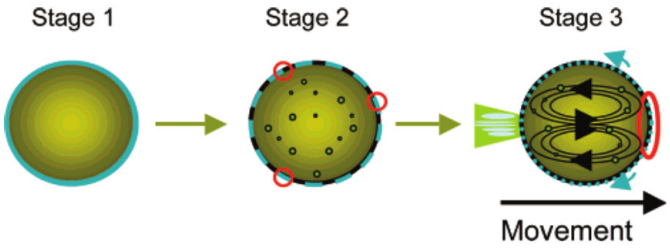


Figure 1.10: Self-propelled nitrobenzene droplets. Reprinted with permission from [Hanczyc et al., 2007]. Copyright 2007 American Chemical Society.

A very recent publication also showed self-propelled oil droplets ([Banno et al., 2016]) using a similar system. They used droplets containing an "inactive" component that was transformed into surfactant and "active" components at low pH. Initially the "inactive" component was hydrolysed, and the resultant molecule protonated into a surfactant at low pH. The new surfactant aggregated at the droplet interface, and generated a gradient of surface tension which created internal flows as in the previous research, moving the droplet forward. What is very interesting about this report is that the components used also created giant vesicles. Although we discussed before that it is very unlikely that droplets were vesicle precursors, it is very interesting to see new research where a droplet is activated through elements in the aqueous phase, moves, and then creates vesicles.

Simple Marangoni flows like the ones explained in Section 1.2.3 have also been used to move droplets containing a big quantity of surfactant molecules. In [Nagai et al., 2005], for example, the authors achieved motion just by using pentanol droplets in a water phase. With respect to the work described in this thesis, this is perhaps the most important research from this section.

The other main way of moving droplets is by using *chemotaxis*, which generates movement as a reponse to a chemical gradient or *stimuli*. This is similar to how natural cells move using their chemoreceptors.

The reference research using chemotaxis is the work from the Grzybowski group, where the authors showed droplets solving a maze ([Lagzi et al., 2010]). The maze was built using an inert material, and it contained an aqueous solution (pH 12). At the "goal" of the maze an acidic gel (HCl pH 1.2) was placed. The droplets contained a mixture of mineral oil and 2-hexyldecanoic acid (HDA), and they were placed at the "start" of the maze. Once inside the maze, the droplets moved following the pH gradient towards the HCl source, and they managed to solve the maze. Their reasoning was that HDA generated a surface tension gradient through the droplet (Figure 1.11). This gradient generated convective flows around the droplet that pulled it towards low pH regions.

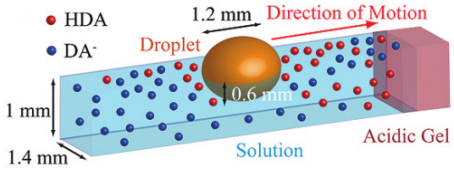


Figure 1.11: Maze solving droplet using a pH gradient. Reprinted with permission from [Lagzi et al., 2010]. Copyright 2010 American Chemical Society.

A more recent research using chemotaxis is more interesting from an abiogenesis point of view because it used prebiotically plausible substances ([Čejková et al., 2014]). They also solved a maze using a droplet, but in their case the droplet was pure decanol, and the gradient was created with table salt (NaCl). The motion described in this system is slower, but it could be manipulated if different salt sources were added at different points in space and time.

There are many other techniques to achieve droplet motion. In [Ichimura et al., 2000] the authors used a light source (UV) to create a gradient in a photoresponsive surface, then the authors placed olive oil droplets that moved following the gradient. While the droplet was moving, by projecting the UV light to another point the authors could re-generate the gradient and move the droplet to different directions. In [Dorvee et al., 2004] motion was achieved by introducing magnetized nanoparticles into a droplet which was then manipulated using a magnetic field. Using this method the authors also managed to fuse two droplets that started a chemical reaction based on their composition. Finally, droplet motion can also be achieved using the mechanical properties of a surface, like its rigidity or stiffness. This is similar to *durotaxis*, where natural cells move based on the mechanical properties of the surface. In [Style et al., 2013] the authors used a lenticular array where they placed layers of soft silicone gel with variable thickness. When a droplet was placed over one of these soft layers with variable thickness, the contact angle across its interface was different, and this pulled the droplet towards thicker regions.

1.3.2 Division

The most relevant droplet division research regarding the results of this thesis is the work shown in [Caschera et al., 2013] where a droplet divided or merged based on a oil-in-water system going from non-equilibrium to equilibrium conditions. In order to get the non-equilibrium initial conditions, a cationic surfactant was placed

in the oil phase (CTAB 20 mM in nitrobenzene), while the aqueous phase contained an anionic surfactant (5 mM decanoate at pH 12). Therefore, when the experiment started and the oil droplet was placed in the water phase, the oil was positively charged, while the water was negatively charged. Both surfactants absorbed to the oil-water interface, one from the inside and the other from the outside, and they flowed through the interface towards equilibrium. This flow decreased the interfacial tension, which caused the droplet to flatten out and become susceptible to physical perturbations. At this point the symmetry was broken, and two ends were created in the droplet phase, pulling its surface towards these ends, until the middle part broke, creating two or more new droplets. The equilibrium point was achieved at 5 mM of cationic surfactant inside the droplet, and 2 mM of anionic surfactant in the aqueous phase. These new droplets did not split, because they were very near the equilibrium conditions. The fusion of droplets was induced with NaCl, which unbalanced the tension at the interface, causing flow and distortion. These disturbances thinned the interface at local spots, causing fusion.

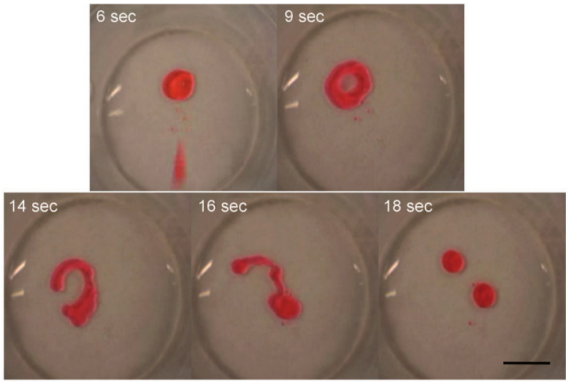


Figure 1.12: Droplet division from [Caschera et al., 2013]. The initial droplet had a volume of 5 μl, while the aqueous phase had a volume of 400 μl. They also reported droplet fusion using NaCl. Reprinted with permission from [Caschera et al., 2013]. Copyright 2013 John Wiley and Sons.

The most popular droplet division system is the one reported in [Browne et al., 2010]. In this case, the two phases were dichloromethane (DCM) with 50% of 2-hexyldecanoic acid (2-HDA) and an aqueous solution of KOH. In this system the surfactant was 2-HDA, which adsorbed to the interface and reacted with KOH, being deprotonated. 2-HDA in its deprotonated form also acted as a surfactant, so it also adsorbed to the interface (Figure 1.13). As the reaction progressed, more and more deprotonated 2-HDA accumulated in the interface, increasing the interfacial area, until the droplet elongated and divided into smaller droplets, where the same reaction continued. The main difference with the research described just before, is that while before the droplets only divided once, in this case they would divide in a continuous way.

A very similar experiment also using 2-HDA has been reported in [Derényi and Lagzi, 2014] where after placing a big droplet of 2-HDA (80 µl) in water, the authors placed inside the oil droplet an aqueous droplet containing a chemical clock pH reaction. This chemical clock pH reaction went from pH 7 to 10, and once

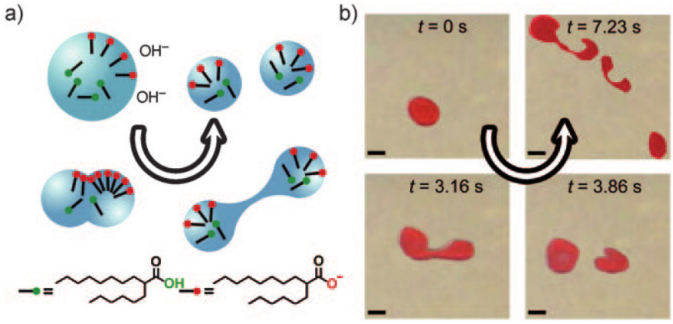


Figure 1.13: Droplet division from [Browne et al., 2010]. While (a) shows a diagram of the chemical reaction, (b) shows video snapshots of the process. Reprinted with permission from [Browne et al., 2010]. Copyright 2010 John Wiley and Sons.

it arrived to 10, 2-HDA molecules deprotonated as before, generating new surfactant molecules, elongating the droplet from the inside and finally achieving droplet division. The interesting bit here is that they managed to only divide the main droplet once by placing the inner droplet, while in the previous research the division happened continuously and was uncontrolled.

The key of the research just described is based on the autopoietic idea of a chemical reaction generating new membrane molecules. Pier Luigi Luisi who is perhaps the leader of the chemical autopoiesis, was the first one to achieve droplet division using the same concept, although technically they were using reverse micelles ([Bachmann et al., 1990]).

The generation of amphiphiles caused by different molecules in the oil and aqueous phase meeting and reacting in the interface is still the main route used for droplet division, as in [Banno and Toyota, 2015]. The interesting difference of this research with respect to the previous ones, is that while before the whole droplet divided into different droplets, in this case the droplet would "bud", until eventually the bud disconnected from the main droplet and created a daughter one.

As with motion, external forces have also been applied to achieve droplet division. In [Song et al., 2014], for example, the authors use a laser to heat a droplet until it split.

Finally, the opposite behaviour, that is, droplet fusion, is actually barely reported in the literature. We mentioned at the start of this section the work of Caschera et al. where the authors achieve fusion using NaCl to disrupt the interface. The only other publication we are aware of is the work described in [Banno et al., 2013] where the authors use oil droplets in a water phase containing gemini surfactants. Gemini surfactants are molecules which are formed by two *regular* surfactant molecules which are joined together usually through their polar head. In said research, depending on the length of the tail of the gemini surfactant, they achieved motion, which is the main behaviour the authors reported, or fusion.

1.3.3 "Multicellular"

So far all the behaviours described consisted of droplets reacting with the medium, like following a gradient or exchanging molecules. The possibility of droplets "com-

municating" with each other is also being explored, which is very interesting from an origin of life perspective in order to study how unicellular organisms became multicellular, or how unicellular organisms communicated with each other to form populations.

A first step is perhaps to study how droplets "attached" to each other. There has been recent research in terms of macroscopic self-assembly through molecular recognition, like in [Harada et al., 2011], where the authors used host-guest chemistry in order to join together blocks of gel in a predetermined order (Figure 1.14). The main difference with respect to all the research already presented is that the authors were not using droplets, but gels. This had the added problem of the gels being static entities, and the authors had to shake the plate to force interactions. This is not completely implausible from a prebiotic perspective, because there could be strong flows that moved the entities around. The main drawback is that gels do not exchange matter with the medium, which is actually what made this research work, because when similar experiments were tried with droplets using host-guest chemistry in order to assemble different patterns, the droplets would expel the molecular recognition units to the medium.

The most similar research to the one just described but using droplets is the work presented in [Hadorn et al., 2012]. The authors used oil in water droplets, but the oil used was diethyl phthalate, which is not a surfactant and is non-polar, but it is more dense than water (1.12 g/cm^3), therefore it formed static droplets, although from an activity point of view they were not that different when compared to gel. Their interesting twist is that the authors used DNA strands for molecular recognition, and this is how they managed to get big "multicellular" assemblies.

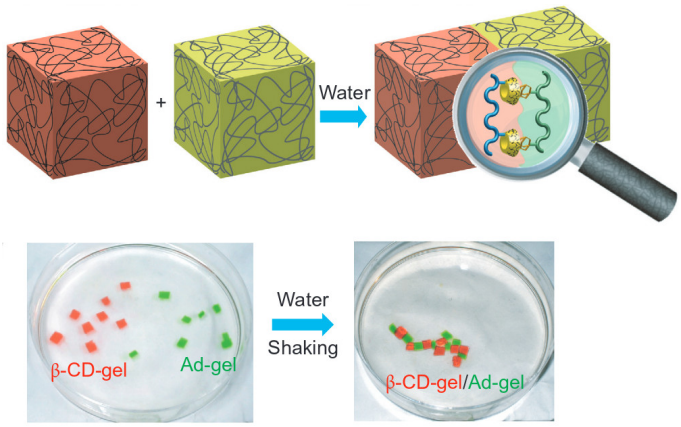


Figure 1.14: Macroscopic self-assembly through molecular recognition from [Harada et al., 2011]. This research used gels, not oil droplets. Reprinted with permission from [Harada et al., 2011]. Copyright 2011 Nature Publishing Group.

Recent research used vapour mediated sensing between droplets in order to have them "communicate" and perform different actions where the droplets were synchronized ([Cira et al., 2015]). The authors used droplets of water with propylene glycol, and placed them in glass plates. Drawing different paths in the plate with an hydrophobic pen the authors managed to get the droplets to move around following each other, fuse or order themselves in a sequence based on their propylene glycol

content. In their publication they explain that the synchronization between droplets was based on a gradient left by the droplets as they evaporated.

Tissue-like materials have also been produced using droplets, like in [Villar et al., 2013], where the authors used an ink-jet printer extruder to place pico-size droplets in a defined structure. The droplets used had a different concentration of salt, and when water flowed across the structure, some droplets would swell or shrink based on the salt concentration, making the whole structure behave like cellular tissue. See Figure 1.15.

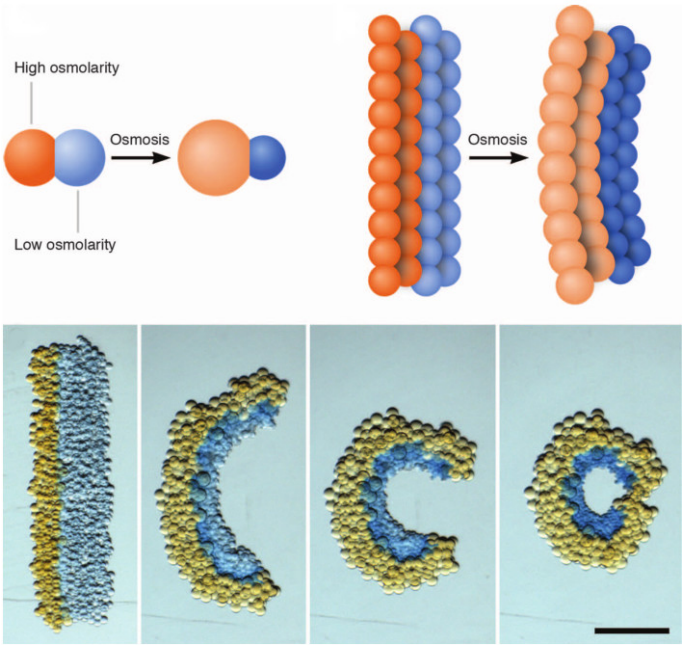


Figure 1.15: Tissue-like material made out of droplets from [Villar et al., 2013]. Reprinted with permission from [Villar et al., 2013]. Copyright 2013 The American Association for the Advancement of Science.

1.3.4 Other *life-like* behaviours

Current protocell research was discussed was discussed during Section 1.1.4 , which in general aims to encapsulate RNA / DNA replication or enzymatic processes, into vesicle-like compartments. These kind of experiments are strictly *life-like* research, because they are directly trying to close the gap between simple compartments and living cells. In this section, though, we will consider a more basic approach based on simpler chemistry which could also be available on prebiotic Earth, long before protocells appeared.

One of the most interesting lines of research with droplets is *morphogenesis*, that is, how emulsions can develop simple morphological characteristics. An example of this research can be seen on [Čejková et al., 2016], which is interesting from a prebiotic perspective because the authors used decanol droplets placed in a solution of alkaline decanoate with 6.5 M NaCl. In order to grow tentacles from droplets the authors let the water in the medium evaporate during a long period of time (Figure

1.16). Initially the droplets adopted a shape with sharper edges, that eventually resulted in spikes, and then finally in tentacles.

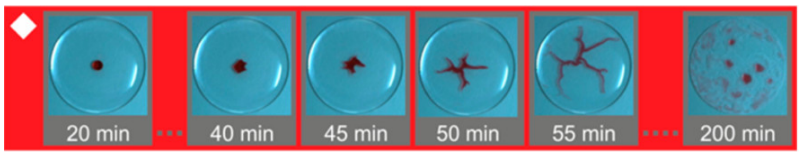


Figure 1.16: Using decanol droplets, and an alkaline medium with a high concentration of NaCl, in [Čejková et al., 2016] the authors managed to grow limb-like structures from droplets. Reprinted with permission from [Čejková et al., 2016]. Copyright 2016 American Chemical Society.

In [Szymanski et al., 2013] the authors managed to couple droplet morphogenesis with a simple internal metabolism. Using an oscillatory Belousov-Zhabotinsky (BZ) reaction inside a water droplet in an oil medium, molecules of the BZ catalyst were incorporated into the membrane during the oxidation stages, that would eventually be replaced by lipids from the oil phase, elongating the droplet.

BZ reactions as a form of proto-metabolism have also been used to achieve droplet motion, like in [Suematsu et al., 2016]. In this case, the oxidized state of the BZ solution had higher surface tension than the reduced state, creating a Marangoni flow (Section 1.2.3) that pushed the droplet forward.

A final example of droplet morphogenesis can be seen in [Denkov et al., 2015]. In this research, the authors used alkane (14 to 20 carbon atoms) droplets containing a 1.5 wt% aqueous surfactant solution. In order to get their results, the surfactant molecule had to have a longer tail than the alkane used, because when the droplet was cooled down, the surfactant that was adsorbed to the interface would freeze first. The hydrocarbons near the surfactant molecules would become plastic crystal, and expand the edges. Depending on the hydrocarbon and surfactant combination, the droplets could present different shapes, like squares, hexagons or different polygons.

1.3.5 Other droplet based research

The field of *unconventional computing*, which among other things tries to emulate a transistor-based computer but without electronics, has also used droplets in order to perform calculations. In [Adamatzky et al., 2012], the researchers used a matrix of interconnected droplets which contained a BZ reaction. The BZ oxidation pulses could transmit between the droplets, and different pulses could start at different points of the droplet-matrix. Depending on the directions from which different BZ fronts arrived to a given droplet, the matrix was able to perform boolean operations like AND, OR or XOR gates.

Continuing with unconventional computing, in [Katsikis et al., 2015] the authors used water droplets infused with magnetic nanoparticles. The droplets were placed in "chips" which were set-up with tiny iron bars similar to a maze. By flipping a magnetic field through the chip, the authors managed to get the droplets moving across it, and based on the maze configuration, and the movement and collision

between droplets, they were able to emulate logic gates and more complicated electronic components, like a finite state machine, a flip-flop or a full-adder.

Droplets have been used in many other different applications, which we will not cover here because these experiments were radically different to what we have described so far. But as examples, droplets have been used as dynamic lenses ([Binh-Khiem et al., 2008]) or as sensors ([Nie et al., 2012]) used to measure blood pressure.

1.4 Evolving droplet experimentation

During the start of this chapter, when the definition of *life* was discussed, it was explained that by most definitions *life* has a strong relation with evolution, to the point that many consider that what defines a *protocell* (first living organism) as being "alive" is its capability to reproduce and propagate its information molecules. From an *origin of life* point of view, perhaps the most popular and accepted theory[21] is the "RNA world", which argues that information molecules started abiogenesis. During the other previous sections, and in particular the one just described about droplet-based experiments, we focused on life-like behaviours instead, ignoring the "evolution" side of life.

There is an on-going debate among researchers about what is more important for life, evolution or autonomy. This dichotomy is covered excellently in [Shirt-Ediss, 2016], where it is explained that the problem resides in the different timescales used. Autonomy explains how living organisms undergo "here and now" processes to be alive, while evolution focuses on how they perform trans-generational information processes so that life can continue after them. Ganti covered this topic during his Chemoton theory ([Ganti, 2002]), where he said that the units of life need to perform actions of two different classes. They need to perform actions to live "here and now", but they also need to perform actions which are not necessary for their instantaneous existence, but required for their self-replication. Ganti said that life lies in the intersection between these two type of actions, and that for example purely evolving chemistry, like replicator molecules, are not alive. This view is not shared among all the researchers, and for example [Chen and Nowak, 2012] opens with "Life is what evolves".

The problem with an evolution centred approach to life is that natural evolution, as it is defined, requires life. Thus, while natural evolution is an unequivocal characteristic of life, it does not help with the process of abiogenesis, which concentrates mostly on how chemistry became alive. We can assume that natural evolution started once the first living organism appeared, but researchers do not agree on what happened before that.

Some of them say that life started after a sequence of stochastic events. They consider, for example, a primordial soup scenario where all the elements were already available, and then a random event, probably associated with a high energy output, like an explosion caused by an asteroid, a lightning, a volcano or tectonic movement, which started a sequence of reactions until biomolecules were synthesised. Some other researchers consider that life came from an outer space body. We will not

[21] Although it has been losing traction for a while.

discuss further about this option, not because it is not a valid one, but because we base ourselves on the hypothesis that life started on planet Earth. Finally, a last group of researchers consider the concept of "chemical evolution". Chemical evolution ([Chen and Nowak, 2012]) aims to explain how the initial four elements after the Big Bang (hydrogen, helium, lithium and beryllium) gave birth to all the other ones in a process known as nucleosynthesis, how this diversity of elements created the first basic molecules, and how these molecules became more complicated up to the point of creating the molecules or life, like lipids or nucleotides. Chemical evolution did not stop with "life", but from an abiogenesis point of view, the moment life appeared chemical evolution became non-relevant and natural evolution took the lead.

1.4.1 Chemical evolution

If we compare natural evolution with chemical evolution, it can be said that natural evolution requires three concepts: self-reproduction, natural selection and variation. As we said during Section 1.1.2, the change from chemical to natural evolution was marked by the transition from self-replication to self-reproduction. Thus, chemical evolution considers self-replication instead. The other main requirement of natural evolution is natural selection, which in the case of chemical evolution can be defined as "dynamic kinetic stability", and its fitness as the "drive toward greater kinetic stability" ([Pross, 2009]). Finally, the last concept required by natural evolution, variation, is fulfilled in chemical evolution the same way it happens in the natural one: mistakes during the replication process.

The main criticism or drawback of chemical evolution is its lack of *heritable information*. Natural evolution has DNA or RNA, or even proto-RNA, and we can assume also that proto-RNA was available during the last steps of chemical evolution, but something before that was required to escape from *stochasticity*. From the lipid perspective[22] of this thesis, the answer to this problem is what is known as "compositional information" or the "compositional genome" ([Segré et al., 2000]).

Segre *et al.* considered the existence of quasi stationary homeostatic[23] states under prebiotic conditions, which they also called *composomes*. A composome could be for example a noncovalent molecular assembly, like a lipid aggregate. The compositional information of a lipid aggregate is defined as a vector describing its exact lipid molecules. Such systems are considered homeostatic if they can conserve their composition through long periods of times, including growth and division. Thus: (a) their lipid vector is their information, (b) they can undergo homeostatic splitting, partially preserving their composomes, and (c) as we said partially, this adds possible mutations and variations. Therefore, such systems are able to replicate and generate offspring with a similar composition and the possibility of mutations, and they undergo processes similar to natural selection, like the drive to higher kinetic stability described before.

[22]This does not mean this is a "lipid world" thesis, we consider that the three sub systems were equally important, and all of them had to evolve probably in parallel until they met into a protocell.

[23]From Wikipedia, homeostatic: A system in which a variable is actively regulated to remain very nearly constant.

1.4.2 Genetic algorithms

During Section 1.3, it was discussed how researchers managed to emulate life-like behaviours in droplets. In order to do so they followed an empirical approach, based on their own observations or on existing literature. During abiogenesis, similar behaviours had to appear due to chemical evolution. During this section we will discuss one of the most common algorithmic implementations of "evolution" with the objective of discussing if such an implementation could work with droplets in order to emulate chemical evolution.

Genetic algorithms (GA) ([Holland, 1975]) are probably the most popular and used algorithm within the family of "evolutionary algorithms", which at the same time are a subset of "evolutionary computation". This section will concentrate on GAs because they offer a simple implementation of the three main evolutionary steps: reproduction (or replication), selection and variation. Nevertheless, GAs have received criticism for not being true representatives of natural evolution[24,25], mostly due to their simplicity, and other implementations based on the same principles have been offered ([Banzhaf et al., 2006]).

In order to explain how a GA works[26] the three main concepts of the evolutionary theory: diversity, reproduction and selection; will first need to be discussed. These three concepts work around an information molecule, like DNA, which describes the *genotype* of an individual, and how this molecular information is ultimately physically represented in the environment where the individual resides in. The physical representation is called the *phenotype*. When one or more individuals *reproduce*, their genotypes are randomly merged together in a single DNA molecule. Human DNA, for example, has around 3 billion bases, and each base can have four different values[27]. Therefore, just by randomly combining two different DNA molecules the number of different possibilities is practically unlimited. Moreover, when two DNA molecules are recombined, the cell machinery responsible of replicating the sequence can make mistakes, which are theoretically random, adding even more *diversity* into the system. These mistakes are called "mutations". Once a new DNA molecule has been created through reproduction[28], there are two *selection* steps. The first one tests that DNA molecule instantly to see if it has all the genes required for life, and if all of them have valid values. The second step tests the phenotype described by the genotype. This test is usually performed during the life-time of the individual,

[24]From now on, natural or chemical evolution will be the same for us. We will continue using natural evolution because it is the most common one.

[25]Being probably the most common artificial intelligence tool, going head to head with neural networks in terms of popularity, and also because their implementation is very simple, they have been criticized for almost everything they do, and more complex solutions have been offered for each possible application. Continuing with the focus of this thesis, we will only be concerned about how they relate to natural evolution.

[26]There are many excellent introductions or reviews to evolutionary algorithms, and genetic algorithms in particular. This section only tries to provide a very basic introductions for readers who might be new to it. If more information is needed, we recommend the first chapter from [Floreano and Mattiussi, 2008] which provides a good introduction for people without Computer Science background.

[27]A (adenine), C (cytosine), G (guanine), T (thymine).

[28]Or self-reproduction.

and in the case of natural selection, it can be considered as the number of offspring generated.

The objective of a GA is to implement these three concepts (selection, reproduction and variation) in a discrete mathematical form, so that a computer can simulate them. Figure 1.17 outlines the main GA steps. It can be considered that what before was reproduction and variation is now encapsulated into a single operation, while a new operation appears with the name of "evaluation". This new operation is now required because while natural evolution does not have a defined fitness function, apart from the loose "number of offsprings", a GA has a well defined fitness function[29], against which the individuals are evaluated. In order to explain how a GA works, we will use as example the particular implementation we used during our experiments, see Figure 1.18 for an overall description[30].

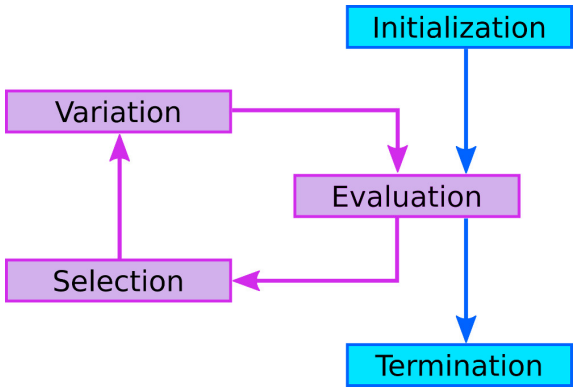


Figure 1.17: Genetic algorithm diagram. While the initialization and termination steps are only executed once, the other three steps are iterated through generations.

The first thing to define is the *genotype*, which in our case was an oil mixture formulation. We used a different gene for each oil, and the actual value of a gene represented the ratio of its related oil into the final mixture. Therefore, the genotypes as described were volume independent, because each oil was assigned a value equal to the percentage it represented in the final volume, with the sum of all oils equal to "1". An example of genotype as used in our experiments could be (0.15, 0.55, 0.20, 0.10), meaning that oil A would represent 15% of the final volume, oil B 55%, oil C 20% and oil D 10%. If the mixture's total volume was 100 ml, each of those percentages would be the same quantity in millilitres. How the genotype is "coded" in the computer implementation can vary. In our case we used a binary representation during the first project, thus a "55%" would represented by its binary code (110111), and a cardinal representation during the second project, where "55%" was just "55".

The next thing to determine is the genotype to *phenotype representation*, which in our case was an actual experiment where four droplets from said mixture were placed in a petri dish already filled with a defined aqueous phase. The kinematic

[29]In most cases. There are other possibilities which will not be covered here but are covered for example in [Floreano and Mattiussi, 2008].

[30]Or Section 3.2.4 (p.78) for the actual implementation of the Dropbot project, and Section 5.4.2.1 (p.134) for the actual implementation of the Flowbot project.

behaviour of the droplets in that system was our phenotype.

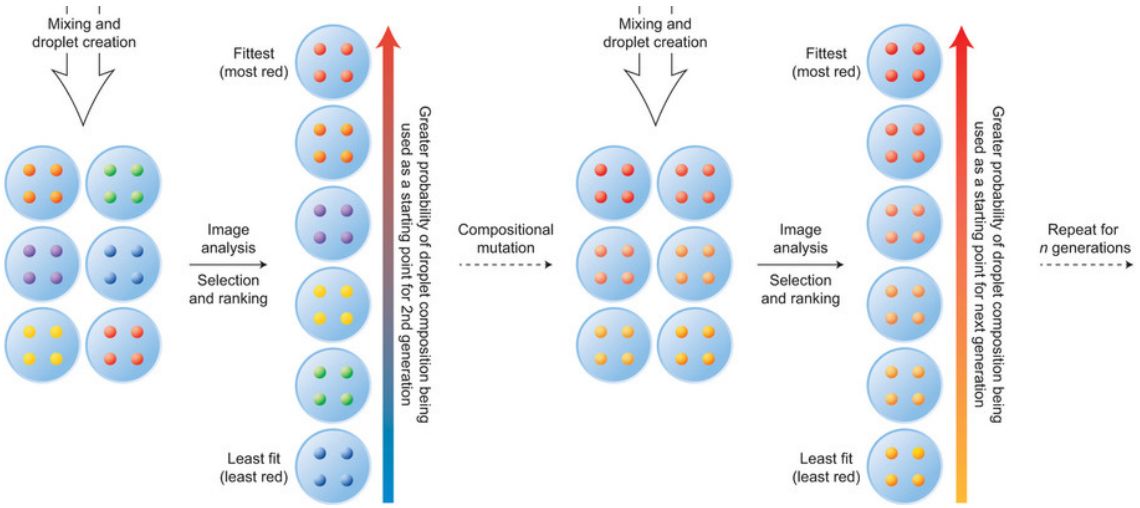


Figure 1.18: GA implementation of a possible oil-in-water droplet experiment. Reprinted with the permission from [Bissette and Fletcher, 2015]. Copyright 2015 Nature Publishing Group.

Once the genotype and phenotype representations are set, the next objective is to define the *initial population*, which on one hand means setting a population size (the number of individuals) and on the other hand means assigning an initial genotype to each individual, which in our case was assigned randomly.

The last thing to define is the *fitness function*. A fitness function aims to *evaluate* a phenotype based on a set of defined characteristics to be optimized. In Figure 1.18, for example, the fitness function is defined as the colour of the droplets resulting from the set formulations, giving higher values to populations with a higher red component. During the experiments of this thesis the fitness functions measured different kinematic behaviours, like movement or division.

Once the initial population has been tested against the fitness function, the next step is to *select* the "best candidates" from it. In order to do so the concept of *evolutionary pressure* is important, because it indicates the percentage of individuals that will create offspring for the next generation. Generally, it is not desired for all the individuals to be part of the next generation, but at the same time just directly choosing the top "n" individuals risks that the population might lose diversity and converge to a local minimun of the search space because the initial population barely covers it. Most selection mechanisms use a randomize factor against the fitness of the individuals. The method we used is called "proportionate selection" and it is based on the next equation:

$$p(i) = \frac{f(i)}{\sum_i f(i)} \tag{1.1}$$

Where $f(i)$ is the fitness value of individual i. In the example shown in Figure 1.18, it would be a value describing how red were the droplets from the i experiment. The left part of the equation, $p(i)$, explains how likely is an individual to be chosen as parent to generate offspring. A common way to implement this equation is the

"roulette wheel" algorithm, that emulates a roulette wheel in a casino, where the size of the slot for each individual is represented by its probability. Therefore, individuals with a higher probability have a higher chance of being chosen. Figure 1.19 shows an example with only five individuals. In order to choose the next generation, we can imagine a casino ball being thrown into the wheel. Individual "C" is the most likely to be chosen, but it might not be chosen at all.

Individual	Level of red	p(i)
A	28	0.18
B	35	0.22
C	76	0.49
D	2	0.01
E	15	0.1

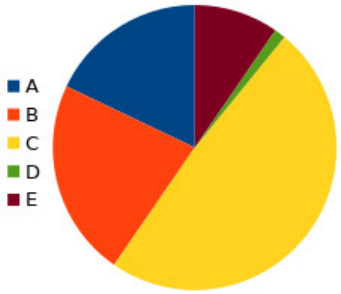


Figure 1.19: Proportionate selection example based on the roulette wheel algorithm.

Once a number of parents have been *selected*, the next step is to build the next generation based on them. The number of chosen parents is related to the evolutionary pressure, which not only depends on theoretical models, but also on possible underlying phenomena from the system. In our case we chose as parents 50% of the previous population, which is a high number in terms of GAs, but our system was based on a very noisy real experiment, thus we wanted to spread our genotype pool as much as much possible.

In order to build the next generation, the parents go through *genetic operators* to create the offspring. The most common ones are recombinations and mutations. A *recombination* merges randomly the genotype from both parents. The most common recombinations are "one-point", where a fixed point is chosen in both genotypes, halving it into two, and each half of a genotype is merged with the other half of the other parent, or "uniform", where each gene is chosen randomly from each parent, see Figure 1.20. The *mutation* operator depends on the genotype representation. It chooses randomly a set of genes, and applies a random drift. If the representation is binary, the mutation can apply a "not" operator to the bit, if it is real, it can apply a gaussian noise.

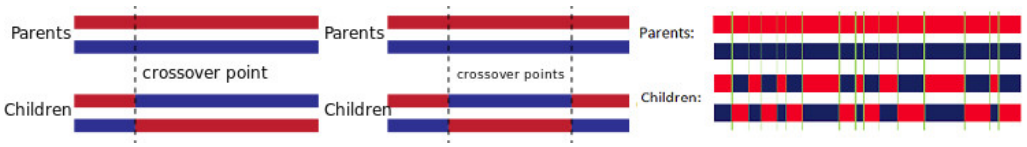


Figure 1.20: Crossover operations. From left to right: one point crossover, two point crossover and uniform crossover. Images from Wikipedia, under CC license.

Once all the offspring are generated, the final step is to build the next generation, which can be fully composed only of offspring, or it can also include some of the previous parents. From a natural selection point of view, it would make sense to

build a new generation only using the offspring, but in our case, for example, because our system was very noisy, we wanted to push parents forward in order to test them again, because it was possible that a given genotype would output two different phenotypes. GA implementations rarely aim to be a pure emulation of natural selection, and a lot of non-natural modifications are applied in order to improve the results obtained.

Once a new generation of genotypes has been created, its individuals would be tested again to obtain their phenotypes and their fitness values, and with this information a new generation can be created, and iterate through this process as explained in Figure 1.17 with the loop between *evaluation*, *selection* and *variation*.

The last step is to *terminate* the execution. In our experiments, a finite number of generations was set beforehand, after which the GA would automatically stop. Some other metrics might compare the improvement in fitness through generations and stop the GA once the improvement is under a set threshold, or set a global fitness value at which to stop once the GA arrives at it.

There are many different implementations of evolutionary algorithms, and many different ways to implement the steps described in Figure 1.17. Herein, we only explained our particular implementation as a way to introduce how a GA works, if the reader wants obtain more information, there are many tutorials covering several of these topics, like for example [Floreano and Mattiussi, 2008].

1.4.3 Evolutionary algorithms applications

During the 60s and 70s, computer scientists and engineers were inspired by natural selection in order to solve problems which were too difficult to solve with analytical methods. Their objective was not to emulate evolutionary processes from an abiogenesis point of view, or to evolve living forms. Their objective was to solve *any* problem, including the two just stated, covering almost any academic discipline. This section will concentrate on real world physical applications of evolutionary algorithms (EA), also known as "the evolution of things" ([Eiben and Smith, 2015]).

One of the most popular examples of a real world physical application is the work described in [Hornby et al., 2011], where engineers from NASA used an EA to optimise the design of an antenna that was going to be sent into outer space and needed to communicate with ground radio using a technology similar to mobile phones. The main difficulty of the task from a human design point of view was taking into account all the complex electromagnetic interactions that could not be described in an analytical way. One of the best antenna designs can be seen in Figure 1.21, which used shapes never considered by human designers. Moreover, after a team of engineers worked for five months on the same project, the EA's design had better gain on a wider range of orientations and a lower power consumption. The EA evaluated the designs in a simulated environment, and only the best ones were prototyped in the real world.

Going from electronics to chemistry, one of the best examples of use of an EA is the work described in [Kreutz et al., 2011] where a genetic algorithm was used to search for new catalysts (See Figure 1.22). Each individual had three different genes: a catalyst, a cocatalyst and a ligand. In order to build their populations,

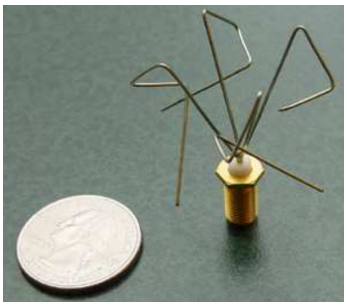


Figure 1.21: Photograph of a prototype of an evolved antenna. From [Hornby et al., 2011]. Image from Wikipedia, released under public domain.

the authors used a well-plate with 48 wells where each individual was assigned to a single well and its components mixed together. The individuals were tested using a microfluidic device[31] using a methane oxidation reaction, and the analysis was also performed on-line using a colorimetric indicator. Based on their fitness values, some of the individuals were selected for the next generation using genetic operations as defined in the previous section. When comparing this research with the previous one, this work implemented real world evaluation, although it still needed a computer to do all the GA operations.

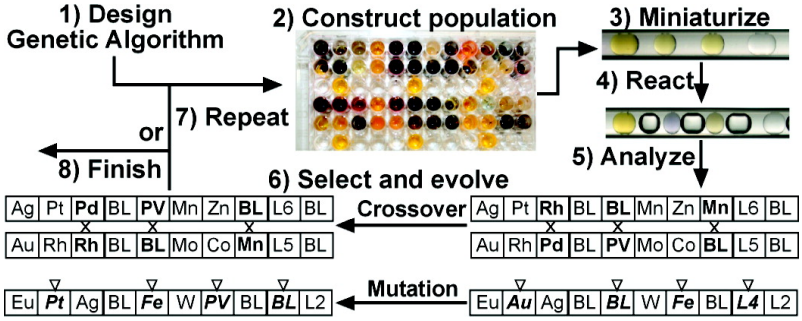


Figure 1.22: Using a genetic algorithm and a microfluidic device as a testing platform, [Kreutz et al., 2011] achieved a search for new catalysts. Reprinted with permission from [Kreutz et al., 2011]. Copyright 2011 American Chemical Society.

From an artificial life point of view, the most relevant work using EAs is the research presented in [Sims, 1994b], where the author introduced the evolution of virtual creatures. The virtual creatures where simulated in a 3D computer generated environment, where for each creature its morphology and the control system for controlling the muscle forces were part of their genotypes, therefore both *body* and *mind* were evolved using a GA. The morphology was represented as an hierarchy of articulated three-dimensional rigid parts, and the control system as a combination of sensors, neurons and effectors. Given an individual, the fitness functions tested were "swimming", "walking", "jumping" and "following". In a follow-up publication using a similar system, the virtual creatures were also evolved to compete with each other ([Sims, 1994a]). Although this research happened completely in a simulated

[31] More about this technique in Section 1.5.2.

environment, a few years later a very similar research was done where robotic lifeforms were evolved in a similar way, and then the best designs were 3D printed into successful working prototypes ([Lipson and Pollack, 2000]).

Although the research developed by Karl Sims supposed a breakthrough in terms of pairing an evolutionary algorithm with the creation of *lifeforms*, the premises upon which it based the design of such lifeforms was very complicated from a prebiotic point of view[32]. A more simplified version of such creatures has also been described, where a creature is formed by several cubic units, being some of the cubes hard and acting like a bone, some of them soft and acting like a tissue, while some acted like muscle and went through contraction and expansion cycles ([Cheney et al., 2013]). Using a GA to evolve combinations of these cubic units, the authors managed to design soft robots that could move, and perform some other actions. Moreover, they also managed to 3D-print the designs using materials similar to the properties used in the cubic units of the designs ([Hiller and Lipson, 2012]). This research can be considered more interesting from a prebiotic point of view, because it showed how very simple aggregations of units with basic different properties could achieve important life-like behaviours such as motion.

Finally, although unrelated to EAs, perhaps the most popular evolutionary experiment is the work described in [Barrick et al., 2009][33] which shows some of the results that Lenski *et al.* obtained in their long term evolutionary experiment which has been running for almost 30 years. This experiment started in 1988, with 12 initially identical populations of asexual *Escherichia coli* bacteria, which as of today have developed into more than 65,000 generations. The main objective of this research is to track genetic changes that may appear through the dynamics of evolution. The importance of their results resides on the fact that the authors validated in real experiments with living organisms the theory of evolution, which is near impossible to study otherwise during a human lifetime, given than evolution works in a much bigger timescale. They showed how successive generations of a population increased their fitness value, and how mutations and recombinations created new genotypes and their respective phenotype representations. Their best result reported so far explained how one of the 12 populations of *E.Coli* evolved to use the citrate present in the medium, while *E.Coli* are theoretically unable to grow aerobically on citrate. Therefore, they managed to show how, through evolution, a population of living organisms could adapt to a new medium.

1.4.4 The role of the environment during evolution

One of the main focus of this thesis is the study of how the environment can shape the *evolutionary dynamics*[34]. By evolutionary dynamics in the context of this thesis it is meant how different population statistical values, such as average genome, average fitness value, or population variability; change through generations. It was just described how in the long-term *E.Coli* experiment researchers managed to get

[32]Karl Sims never aimed for his research to be prebiotically plausible.

[33]And many other publications about the same subject from the same research group

[34]The relation between environmental variation and evolutionary dynamics was the main motivation of the Flowbot project (Chapters 5 and 6).

populations of bacteria to adapt to new environments by changing their medium. Nevertheless, it can be said that the impact of the environment or environmental variations on evolution has been under-studied. For example, in the previous section, all the EA examples described considered the problem "in vacuum". Not only that, but most abiogenesis theories, as described during Section 1.1.2 consider "in vacuum" the chemical evolution of the molecules necessary in life: they do consider a primordial soup scenario where everything was available, but they rarely consider how the environment could impact it. An exception, as described during said section, is the theory of the hydrothermal vents ([Lane and Martin, 2012]), and all the other similar research that focuses on plausible geographical locations for abiogenesis. But, as said, it is still an under-explored area, and the primordial soup scenario is still the main framework of work.

With respect to evolutionary algorithms, most of the research done studies the environment as part of the problem and aims to define new algorithm implementations in order to optimize the genotype through it: they consider the environment a passive element of the system, at most a part of the fitness function. In [Jin and Branke, 2005], for example, which is the reference survey for evolutionary algorithms in terms of how to deal with "uncertainty", the authors considered the environment as everything that is not a "design variable", and they explained that in order to find an optimal solution there are three strategies to cover: noise, robustness and fitness approximation. In particular, they deemed the environment very related to the robustness of the solution, and they developed a series of strategies in order to generate a solution that could be optimal for a set of different environments. In a similar way, in [Cobb, 1990] it is studied how by increasing the mutation rate after each environmental change its effects on evolution can be eased. Or in [Grefenstette, 1992], where the authors recommended to maintain population diversity in order to overcome changes in the environment. Finally, a more interesting example because it is applied to a real life problem while the previous ones were mathematical simulations, is the work in [Bredeche et al., 2010], where the authors considered a population of agents placed in a unknown physical environment that could change dynamically. They authors proposed an algorithm that would be distributed to each agent, where each agent competed with each other, but that eventually would arrive to a consensus about the best solution. The authors described how this consensus strategy generated more robust solutions against unknown or changing environments.

From an abiogenesis point of view, natural evolution is considered a constant process that does not *tweak* itself in order to overcome environmental changes, apart from a constant spread of genotypic information. Thus, instead of trying to re-implement our algorithms in order to overcome environmental changes, we will study the environment as an active actor that shapes the dynamics of evolution, and we will concentrate on how these dynamics add a new dimension into the genotype to phenotype correspondence[35]. It will be considered how the environment *tweaks* evolution, and not how to *tweak* evolution to overcome the environment.

Following this line, some of the most relevant research is the work presented by

[35]It is known that ecology plays a critical factor on the phenotype representation. Our objective is to quantify it.

Alon *et al.* ([Kashtan et al., 2007] and [Parter et al., 2008]). In the 2007 publication the authors studied how by varying the environments through an experiment evolution can be sped up. In order to study it, they compared experiments with a single environment, experiments with multiple environments that were generated randomly, and experiments with multiple environments that could be decomposed into modular related sub-goals (Figure 1.23). They tested the different scenarios against very different problems, going from logic gates circuits to RNA structure, and they showed than when using modular related environments, evolution was sped up from 25 times up to 700 times when compared to experiments that used a single environment, and even when just using random environments, some of the problems were sped up around 50 times.

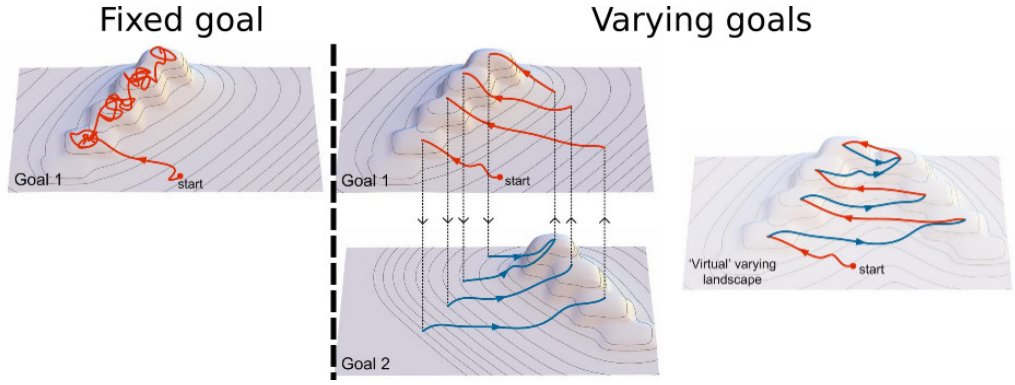


Figure 1.23: Left: Evolutionary trajectory where only one environment was used. Populations tended to spend time in local minima until a lucky mutation pushed them forward. **Right:** By alternating between two environments with a shared a goal, evolution was sped up. Dashed lines mark where the goal was changed. Adapted from [Kashtan et al., 2007]. Copyright 2007 National Academy of Sciences.

In their follow-up paper from 2008, the authors studied how by varying the environments in the same way as described before, the facilitated variation (novelty) of the individuals was enhanced. Their results showed that the organisms evolved through such conditions kept a history of past environments in what they called "genetic triggers" of the genome sequence. These genetic triggers are what helped the organisms to develop novelty and generalize their phenotype to future environments. One of their surprising finds is that populations of individuals that were evolved through several environments had less genotypic diversity that the ones evolved through a single environment. This happened because with more environments used, more parts of the genome sequence were used by genetic triggers, which left less space for individual variation.

The field of *virtual* creatures has also studied the influence of the environment on such creatures, because the platform itself is perfect for creating different environments in which to test populations of creatures against several fitness functions. In [Auerbach and Bongard, 2014], for example, virtual organisms were designed in a similar way to the ones described by Karl Sims as discussed before. Their main difference is that in this research the authors restricted the nervous system in order to focus on the morphology, because they wanted to study if complex environments

would result in complex morphologies. The environments the authors used were flat areas with ridges or canals. The spacing between the ridges and its height were the variables used to define a given environment. Their results showed that as the spacing between the ridges was narrowed such that a creature covered several of them, the creatures had to learn how to move along the tops while finding ways to push themselves using the gaps. A very similar research is shown in [Lassabe et al., 2007], where the authors also based their creatures on the designs of Karl Sims, but they simplified the morphology by only using cuboids. While the previous research only focused on motion, this research extended the fitness functions to several other behaviours, like climbing or skating. The environments described were also more complicated, with trenches, staircases, walls and disconnected pillars where the creatures needed to learn how to jump from one to the other. Soft robots formed by cuboids with different properties have also been studied in similar problems, like for example in [Cheney et al., 2015] where a soft robot was encapsulated in a tight space that it needed to escape. The size of the robot, and the size of the opening gap were very similar, thus the robot had to learn how to compress its soft parts in order to move through.

The relation between environmental changes and evolution is well studied among evolutionary theorists. It is usually considered that organisms can undergo two different processes: phenotypic plasticity or phenotypic diversity ([Xue and Leibler, 2016]). Phenotypic plasticity works on the individual level, and explains how the same gene set can manifest different phenotypes. It is considered that individuals can "sense" the changes and adapt as required. An example of phenotypic plasticity is the seasonal timing of flowering ([Amasino, 2010]). Phenotypic diversification acts on the population level, and considers how the phenotypic pool of a population is wide enough to survive an environmental change. An example of phenotypic diversification is how populations of bacteria survive against antibiotic ([Balaban et al., 2004]). An organism and population can use these two strategies.

From a biology perspective, the relation between the environment, evolution and development is studied in the field of "eco-devo-evo" ([Abouheif et al., 2014]), which is an update of the "devo-evo" model which considers evolution as the result of inherited changes in development. Eco-devo-evo expands this by stating that the environment induces genotypic and phenotypic variation, while the development is a regulator that can mask, release or create new combinations of variation. Finally, natural selection takes this variation, and adapts it into new genotypes. These concepts have been been applied to virtual creatures, like in [Arita et al., 2016]. In their model, the organisms grew from a single cell through multiple divisions until they arrive to a larva stage, at which point the larvae were placed in an aqueous environment and left to evolve. While evolving inside the aqueous phase, and through their own development, the creatures arrived to its adult stage, and from that point on they were evolved in a land environment. Thus, the morphology of a creature was evolved to provide a solution for two different environments.

Finally, although unrelated to evolutionary algorithms, the relation between evolution and the environment has also been studied in natural examples outside the lab. A recent example of such research is [Ghalambor et al., 2015], where the authors focused on the concept of phenotypic plasticity, which explains how the same

genotype can produce different phenotype responses based on environmental variation. In order to do so, the authors took a type of fish (trinidadian guppies) which are adapted to living with cichild predators, and transplanted them to new streams where this predator was not present. After four generations they found 135 genes that evolved changes into the predator-free population, but 89% of these changes were not adaptive. Thus, the genetic plasticity was often reversed in the following generations. An example of the relation between evolution and morphology can be seen in [Passy, 2002], where the morphology of a type of algae was studied and they found a relation between more complex morphology in more complex environments. The authors considered that a complex morphology could be a survival strategy in unpredictable environments. There is a last factor which is rarely studied in simulation, and that is the "eco-evolutionary feedback": not only living organisms adapt to an environment, they also change it, and this change needs new adaptation ([Post and Palkovacs, 2009]). A similar concept to eco-evolutionary feedback is the one of "niche construction" ([Laland and O'Brien, 2010]), which focuses on how a organisms can modify natural selection in their environment and act as co-directors for their own evolution, as well as the evolution of other species sharing the same environment.

1.4.5 Open-ended evolution

If an evolutionary algorithm is intended to emulate natural or chemical evolution, it must be open-ended. [Taylor, 2015] defined open-ended evolution as "evolutionary dynamics in which new, surprising, and sometimes more complex organisms and interactions continue to appear". The author also stated that open-endedness on artificial systems requires five fundamental properties:

1. Individuals should be able to robustly produce at least one new robust individual.
2. The medium should allow the existence of unlimited diversity of individuals.
3. Individuals should be able to produce offspring of higher complexity than themselves.
4. There should be several mutational pathways within the fitness landscape.
5. There should be a continued drive. The individuals must experience a changing landscape.

As Tim Taylor reported on his review, although several systems have been designed to be *open-ended*, none of them has fully achieved that. Perhaps the most famous example of open-ended research is the Tierra project ([Ray, 1991]), which is a computer simulation program where entities compete for CPU time and memory space. These entities can mutate, recombine and evolve. The main difference between Tierra and other evolutionary algorithms, like genetic algorithms, is that Tierra does not have a defined fitness function: the entities fight for survival, similar to natural selection. Based on this idea, Tierra was considered an open-ended

candidate framework for a while, but statistical analysis showed that it does not fulfil all the requirements ([Bedau et al., 1997]). Tierra's main limitation is that its organisms are hard-wired to isolated portions of memory, and they do not have write access to other parts. Moreover, the virtual CPU which governs the process does not evolve, and that is also considered a requirement for open-ended evolution.

1.5 Automating droplet experimentation

During Section 1.3 it was explained how droplets can be models of lipid aggregates that could eventually become a protocell, and during Section 1.4 it was explained how evolutionary algorithms can be used to emulate, with some limitations, natural evolution. Thus, it would be interesting to use an evolutionary algorithm to optimize or evolve droplets, from basic lipid aggregations to ones possessing life-like behaviours like the ones described previously. Evolutionary algorithms, though, and in particular genetic algorithms, require a big number of evaluations or tests in order to evolve a population of entities. It would be possible for a person to execute all the tasks manually, but considering that thousands of experiments might be required, automating it might be a more efficient option. Also, apart from pure automation, robots allow for a task to be repeated under the exact same conditions, every time, as opposed to how a human would do it, which would inevitably add small variations between each experiment.

This section will review the most common platforms used in order to generate droplets. In particular, the focus will be on platforms that can generate droplets of size similar to the research described before (Section 1.3) which had volumes between 1 and 5 µl on average. These volumes are not a technical limitant factor, because automated lab equipment can generate droplets up to the pico level.

1.5.1 Liquid handling robots

All the experiments described during Section 1.3, except one, where an inkjet printer was used, were performed manually using either a handheld syringe or a handheld pipette. These handheld instruments are very common in any life science lab, and there has been a lot of advances giving these handheld tools high precision to the point that they can easily work with the volumes just described. Their main limitation is that because they are operated by a human researcher, their level of high throughput is limited. Even though there are plenty of multi array pipettes that can generate several experiments at the same time, doing thousands of manual experiments with them is not an option.

If we study how a human would develop droplet based experiments, it can be said that liquid handling robots (LHRs) are the platform to choose ([Kong et al., 2012]), because they automate human-like action using mechanical and electronic operations. Specifically, LHRs focus on *liquid dispensing*, which is the action a human would do when actuating a syringe or pipette, *positioning*, which is the action that describes where exactly the liquid is placed, *washing*, because equipment needs to be cleaned constantly in order to perform more experiments, and *sensoring*, which

in the case of a human it would be just to check that the experiment was performed in the correct way.

Liquid dispensing is most important action for a LHR. Therefore, it is the one that has been engineered to be more precise and robust, and while some of the other actions are shared with other types of robots, liquid dispensing is almost unique to them. The main problem with dispensing liquid is getting the actual liquid released from the tip of the instrument, because when working with small volumes surface tension is stronger than gravity, and the liquid will not be released by itself. In most LHRs, the liquid dispensing is performed either by contact or non-contact actions. Contact methods use a rigid structure, like a pin, where the liquid is placed on its tip, and then the tip touches the surface of a substrate, and then a drag-back action is used to break the surface tension between the liquid and the tip. Non-contact methods are exactly what syringes or pipettes do, where a mechanism pushes the liquid out. There is also a mixed option where a non-contact method flows the liquid to the tip, and then the tip contacts a surface where the adhesion of the fluid is broken, pulling the droplet away from the tip. The main non-contact methods used are based on inkject printers, and they use a solenoid valve, or a piezoelectric. Solenoid valves are used for bigger volumes, and piezoelectric for smaller volumes.

Moving the dispensing tool around the workspace is the second priority of a LHR, but it is a well studied technology in many other robots, like Computer Numeric Control (CNC) machines. The most common frame structures used are gantry or cantilever, where with a series of motors, pulleys and belts the tools are moved around (see any of the robots in Figure 1.24), although robot arms stations are also used ([Fleischer et al., 2016]). The washing mechanism is also critical, because contamination must be avoided at all cost. Its implementation depends entirely on the experiments being performed, but LHRs will usually guarantee that the dispensing tools are always clean and ready after its use. Finally, most LHRs use a series of sensors to monitor the actual actions of the robot. These sensors can be "defensive" mechanisms that will check if the dispensing or moving actions were performed correctly, or they can be "experimental sensors", which will feedback data from the experiments to the user or control computer.

Considering these main four points that a robot tries to solve, there are two different platform solutions. On one hand, a workstation can be used that will automate everything, or on the other hand, this process can be separated into different automated equipment, in which case the different equipment will need to be interfaced, either by another automated platform, or by a researcher. Chapter 3 focuses on how a workstation robot was built from scratch as part of the work developed during this thesis, where the concepts just described will be covered in more detail.

The state-of-the-art of liquid handling robots can be found in the industrial and commercial world. Herein, it will be reviewed some of the commercial robots available that would be suitable for droplet experiments as the ones described.

"Hamilton" is a company specialized in LHRs. They have two lines which might be suitable for the experiments described: Star ([Hamilton, 2016b], Figure 1.24 A) and Nimbus ([Hamilton, 2016a], Figure 1.24 B). The *Star line* is said to be based on a modular design, where an initial base robot is purchased, and then different upgrades can be applied in order to achieve different objectives. Similarly, they also

sell in a modular way all the relevant software. The robot can use different pipetting heads, some of them having around 100 different pipetting channels, while a basic model has from one to four independent air displacement channels. The tips of the pipettes are dispensable. The lowest volume it can handle is 0.5 µl and the biggest is 1 ml. Although they do not disclose if they use a contact mechanism to generate such small volumes, or if it is just the liquid displacement the system can achieve. It can handle 5 µl with an error of a 2%. The robot is said to be able to have different applications like benchtop genomics or cell culturing. The *Nimbus line* is a slightly simpler robot. The model "NIMBUS4", which is the lower end one, has six channels. Its lowest volume is 1 µl, although the accuracy is not disclosed, but it only has 1 channel with this precision. The accuracy of the other five channels is 50 µl.

"Hudson Robotics, Inc." is a company specialized in micro plate automation and liquid handling. They have two models of LHRs: Solo ([Hudson-Robotics, 2016b], Figure 1.24 C) and Micro10x ([Hudson-Robotics, 2016a], Figure 1.24 D). *Micro10x* focuses on nano volumes and uses a contact dispensing mechanism, so we will concentrate on the Solo line instead. The *Solo* robot uses a robotic arm to handle the liquid which uses a pipette to dispense it. It has from one to eight channels. It can handle 1 µl with an standard deviation of 2.72. When handling 5 µl its standard deviation is 1.50. They also sell a series of accessories that can be connected to the robot, like a plate stacker or a reagent dispenser.

"Tecan" is a company that covers a wide range of different applications of life sciences. Their more interesting robot is called EVO 75 ([Tecan, 2016], Figure 1.24 E), which is also a modular robot. It can handle from 1 µl up to 5 ml. The accuracy is unknown. It uses a special type of pipette which the authors claim to be more precise and also automatically cleaned after every use.

"Perkin Elmer" is a general purpose biotechnology company. Their more interesting robot is called "Zephyr Compact Liquid Handler" ([PerkinElmer, 2016], Figure 1.24 F). It has 8 channels using pipettes, and it can handle 0.5 µl with an error of 8%. When handling 5 µl its error is 5%.

"Eppendorf" which is one of the best known companies in terms of laboratory equipment, has a platform in the line of automated pipetting called "epMotion" ([Eppendorf, 2016], Figure 1.24 G), which can handle a lowest volume of 1 µl.

All these robots are extremely well equipped and robust, but their main limitation is their customizability. Even though a lot of them claim to be modular, if something slightly different to what they were designed to do needs to be done, it is usually nearly impossible to sustain a fully automated process. As an example, all these robots offer a software suite in order to program its operation, but in none of the robots is the software stated to be open source. Thus, it is not clear if for example it would be possible to use a genetic algorithm to program the robot operation. This lack of customizability is the main reason why different research groups are nowadays building their own robotic workspaces. Next, we will cover some of the most relevant do-it-yourself (DIY) solutions.

A first strategy is to build a workstation from existing commercial working components. An example of this can be seen in [Peddi et al., 2007], where the authors integrated a commercial liquid handling robotic system called Xantus from the com-

pany *SIAS AG*, with a microprocessor-controlled microsyringe pump from another company, see the whole robot in Figure 1.24 H. The Xantus system had originally two arms, one used for liquid dispensing, and the other used as a plate gripper. The authors replaced the plate gripper with the microsyringe pump. They also had to change the motor that operated the second arm in order to reduce its step size from 100 to 12.5 µm. Finally, because the positioning of the tool was criticial in their research, they also designed a coordinate measuring machine in order to detect the head of the dispensing tool in real time.

Another strategy is to fully build a liquid handling robot from scratch, like in [Faina et al., 2016] (see Figure 4.15, image C, in p.109), where the authors built the robot frame using strut profiles, the moving mechanism using stepper motors, and in order to dispense the liquid the authors used handheld glass syringes actuated with a stepper motor so that the plunger could be actuated to absorb or release liquid. Moreover, the software and electronics the authors used were open source and are publicly available. A very similar open source robot is the one described in [Müller et al., 2015]. The main difference with the previous project is that this one used a 3D printer in order to build the parts (Figure 1.24 I). There has also been recent development in terms of creating only liquid dispensing tools using 3D printers, like the research reported in [Wijnen et al., 2014], where the authors developed an 3D printed syringe pump, and all their research is open source, meaning that it could be added to other open source sources. Finally, while these open source projects are "research" projects, there is a commercial open source liquid handling robot, called "Opentrons" ([Opentrons, 2016]). They sell three different robots, one with a single channel pipette, one with a multi channel pipette, and one that can also culture tissue.

1.5.2 Micro and milli-fluidic devices

Most of the liquid handling robots discussed before can generate droplets because they are designed to handle volumes between the pico and nano levels. Nevertheless, that does not mean that they were designed for droplet-based experimentation, like the type of research described during Section 1.3, where the study focuses on the interactions between oil, water and the interface. Most of the liquid handling robots described focus instead in fields like proteomics, genomics or general cell-based assays and growth processes. The platform more popular nowadays for pure droplet experiments is the microfluidic device ([Casadevall i Solvas and DeMello, 2011]).

A microfluidic device is a platform containing a network of channels connected to a series of inputs, outputs and different sensors. Its particularity is that the channels used have a width lower than 100 µm, and they can even go to the nano level. Microfluidic devices are used in many different displicines, from biology, to chemistry and even physics. Figure 1.25 explains the main methods used in order to generate droplets. These methods are based on having two or more channels with different phases. In the case of a capillary, one channel is inside the other, while in the case of a T-junction or flow focusing, the flows come into contact with each other in a perpendicular way. These two last methods are the most common ones

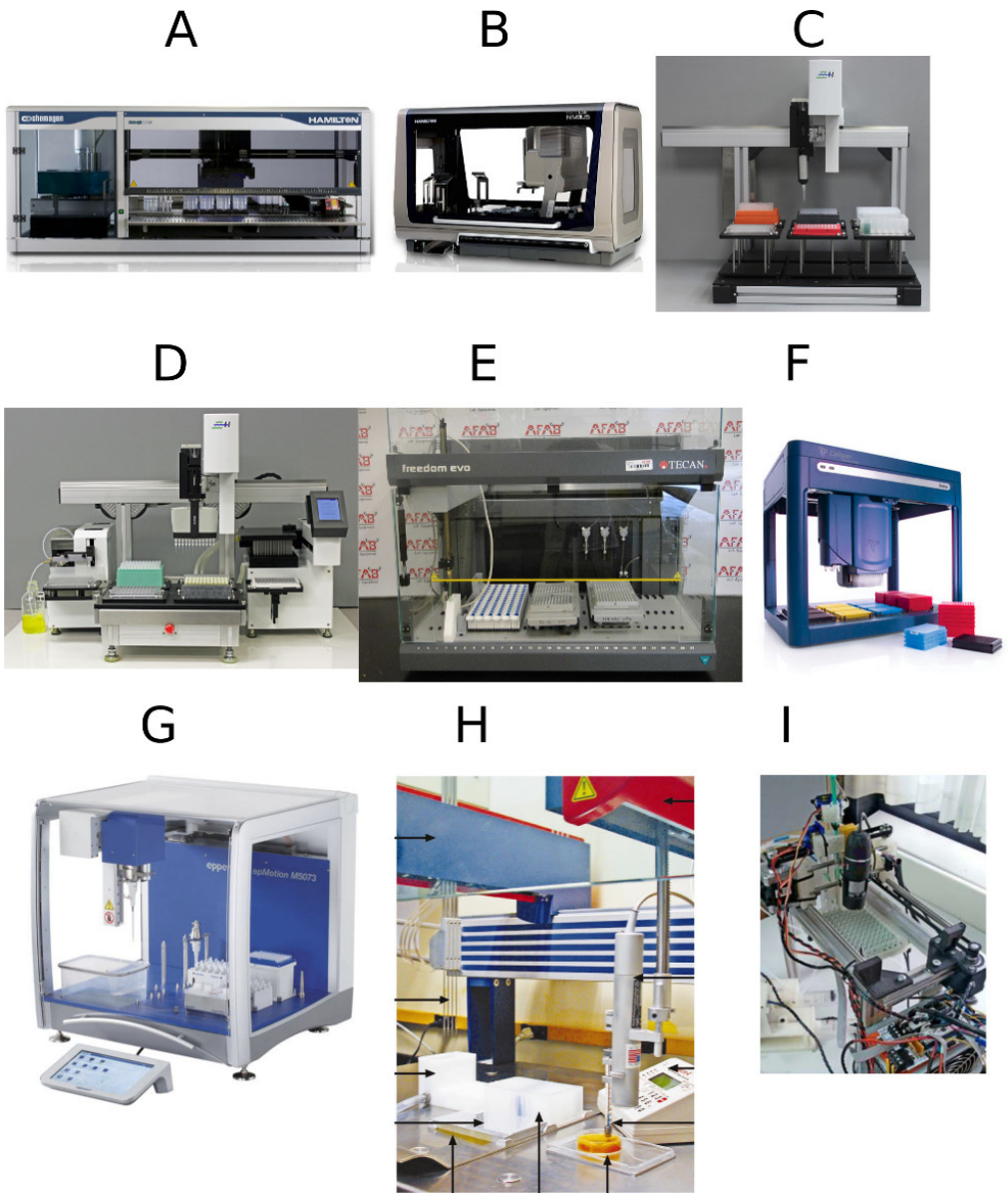


Figure 1.24: Liquid handling robots. **A:** Hamilton Star from [Hamilton, 2016b]. **B:** Hamilton Nimbus from [Hamilton, 2016a]. **C:** Hudson Solo from [Hudson-Robotics, 2016b]. **D:** Hudson Micro10x from [Hudson-Robotics, 2016a]. **E:** Tecan Evo from [Tecan, 2016]. **F:** Perkin Elmer Zephyr Compact Liquid Handler from [PerkinElmer, 2016]. **G:** Eppendorf epMotion from [Eppendorf, 2016]. **H:** DIY robot from [Peddi et al., 2007]. **I:** DIY robot from [Müller et al., 2015].

because they allow for a better control and a higher frequency.

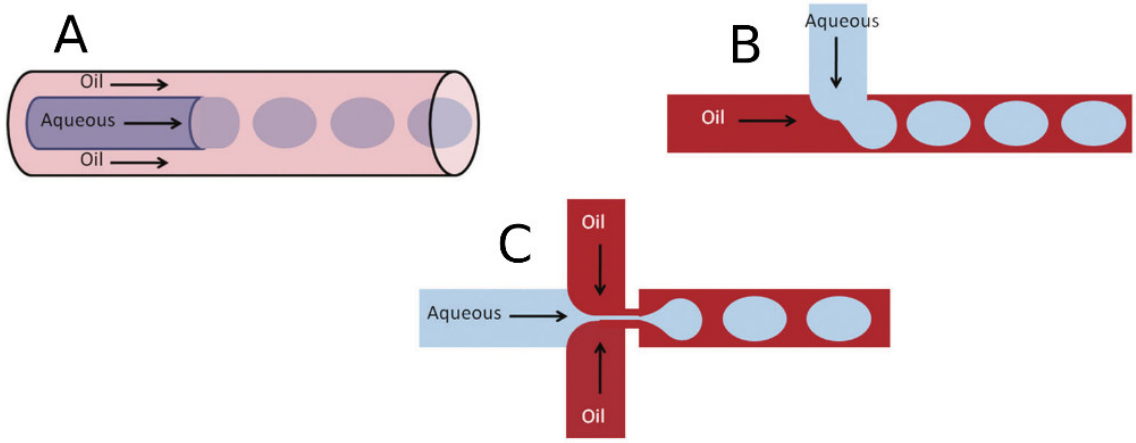


Figure 1.25: Microfluidic devices droplet generation. **A:** Capillary. **B:** T-junction. **C:** Flow focusing. Although the diagrams created water droplets in an oil medium, the phases can be inverted to generate oil-in-water droplets. Adapted from [Casadevall i Solvas and DeMello, 2011]. Copyright 2011 Royal Society of Chemistry.

Although microfluidics are a very good technology, they have two problems, one of them general to the platform, and the other one particular to the chemistry. Their general problem is that they are very difficult to manufacture. Their channels and connections are very small, and flowing perfect liquid at such low volumes is complicated, moreover, any small deformity in a channel can change completely the flow. Their second problem, from a chemistry point of view, is that they are commonly manufactured with polydimethylsiloxane (PDMS). The chemical compatibility of this material is low, and it would swell when exposed to most organic solvents.

In order to tackle the first problem about their complicated manufacture process, some researchers instead build devices with slightly bigger channels, aiming for channels of 0.3 mm or larger instead of at most 100 µm. These devices are not microfluidic anymore, and they are called *millifluidic*. Commercially available fused deposition modeling (FDM) 3D printers based on additive manufacture ([Dimitrov et al., 2006]) can have a resolution as low as 400 µm. Therefore, millifluidic devices can be manufactured using a standard FDM 3D printer ([Comina et al., 2014]).

An example of research in this direction is [Tsuda et al., 2015], where the authors used an "Ultimaker" 3D printer, which is probably the most common commercially available FDM one for *hobbyists*, and they used polylactic acid (PLA) and polyethylene terephthalate (PET) as filaments. The authors were able to manufacture "plug and play" devices where several of these devices were connected to form a bigger device (Figure 1.26, A). The channels the authors designed had a width of 400 µm, and they were able to robustly generate droplets with the same diameter. Not only that, but the authors were also able to add PDMS layers which were used as valves, and the authors also managed to integrate LED components for cell culturing. Similar devices were also fabricated using the same FDM 3D printer in [Yuen, 2016]. In this case, the devices were connected together using magnets (Figure 1.26, B). In a similar way using a 3D printer, but in this case using the technique of stereolithography. In [Bhargava et al., 2014] the authors were able to manufacture millifluidic

devices that could be "plug and play" in order to perform droplet operations similar to an electronic circuit (Figure 1.26, C). For this research, though, the authors used an industrial 3D printer, and the devices were not printed by them in their lab, but by an external company. Finally, another possibility is to build your own 3D printer from scratch, like in [Lee et al., 2015] where the authors built their own 3D printer which used stereolithography in order to manufacture the devices. The smallest channels the authors could print had a width of 300 µm, which is still considered a millifluidic device (Figure 1.26, D).

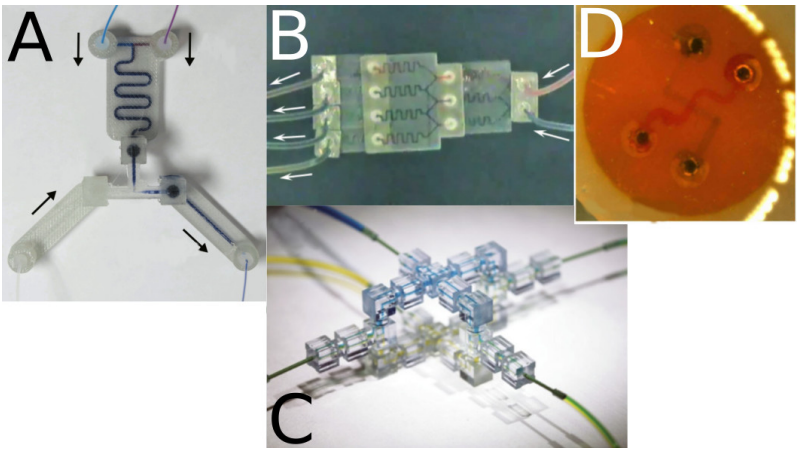


Figure 1.26: 3D printed millifluidic devices. **A:** Commercial FDM printer from [Tsuda et al., 2015] (CC 4.0 license). **B:** Commercial FDM printer from [Yuen, 2016] (Reprinted with permission from [Yuen, 2016]. Copyright 2016 Royal Society of Chemistry). **C:** Industrial stereolithography printer from [Bhargava et al., 2014] (Reprinted with permission from [Bhargava et al., 2014]. Copyright 2014 National Academy of Sciences). **D:** In-house designed and built 3D printer from [Lee et al., 2015] (CC 4.0 license).

It is not only FDM or stereolithography that have been used to 3D print flow devices, some other techniques like laser sintering or selective laser melting have also been used ([Capel et al., 2013]). The main problem of these other techniques is that they are less extended, and the technology itself is under a patent status or very recently released, which means the equipment is less affordable. Moreover, they rarely go under 0.1 mm per channel. The exception is a new technique called "two photon lithography" or "multi photon lithography" ([Jonavic et al., 2016]), which is similar to stereolithography but offers higher resolution that can go as low as 65 nm ([Haske et al., 2007]). Although there have been images of micro or even nano-fluidic devices fabricated with such technology, we could not find anything reported in the literature.

1.5.3 3D-printed *reactionware*

In the different projects just discussed where a 3D printer was used to manufacture a device, they achieved an easier manufacturing process when compared to microfluidic fabrication, although with the drawback of not really fabricating devices on the micro scale. The other problem we mentioned, chemical compatibility of the material, was

not improved because for example the FDM 3D-printer mentioned used PLA, which is biodegradable, while the other two using stereolithography based projects used polyethylene glycol (PEG), which is also biodegradable. The use of these materials limit the chemistry that can be flowed through these devices. In particular, most of these materials cannot work with acids, and one of the base molecules of abiogenesis is a fatty acid (Section 1.2.6).

Cronin *et al.* coined the term "reactionware" to describe the concept of 3D printed devices that could themselves be an active element of a chemical reaction (see [Symes et al., 2012] or a more recent and general protocol-like publication in [Kitson et al., 2016]). This line of research is very relevant here, because in most of their devices they used polypropylene (PP) as manufacturing polymer, which has a high chemical compatibility with many solvents, like acetone, and also to acids, like fatty acids. Of particular interest is the research the authors described in [Kitson et al., 2012], where the authors 3D printed millifluidic devices using PP, although they did not generate droplets. In a follow-up study the authors used similar devices with "flow organic chemistry" ([Dragone et al., 2013]), showing that the devices they fabricated were indeed able to contain a wide range or organic solvents and reactants. Finally, they also fabricated similar devices that were used as hydrothermal bombs ([Kitson et al., 2012]), therefore, using this technology leak-proof devices could also be manufactured.

Although unrelated to "reactionware" or "flow chemistry" the use of 3D printers has been extensively used recently in order to aid lab processes in different ways. For example, in [Baden et al., 2015] the authors defined the concept of "open-source labware", which refers to the idea of creating 3D models of lab equipment, and then manufacturing them using a 3D printer and basic hardware components. They use as an example a micro-pipette the authors designed. Another examples can be seen in [Porter et al., 2016] where another group of researchers designed and 3D printed a colorimeter, in [Philamore et al., 2015] where the authors 3D printed membranes for fuel cells, or in [Grasse et al., 2016] where a group of researchers designed a 3D printed UV-vis spectrophotometer that used a smartphone to analyse samples.

1.5.4 Droplets as artificial life *lifeforms*

During this section so far it has been discussed how to automate droplet-based experiments, either using a full-fledged robot or a flow/fluidic platform. Herein, the problem will be looked at from the opposite direction, and it will be discussed how droplets can be "robots" themselves, and how abiogenesis processes can be applied to the development of artificial life.

Nowadays there are two main conceptual paths in order to create artificial life. One of them, which is the most extended one, aims to design robots based on the technology available at that moment. If necessary, technology is "pushed forward" in order to implement new functionalities. These robots are the mechanical / electronic ones. Independently on if they aim to be humanoids, drones, quadcopters, or robot arms; engineers try to *engineer* an existing design with the technology available, which is the limiting factor. It can be assumed that humanoid robots in 50 years will be nothing like the ones today, simply because the technology will be more

advanced.

On the other hand, abiogenesis based artificial life is not limited by the technology available, but by the lack of knowledge or description about the exact processes that started life. Droplets are interesting artificial life candidates because they easily manifest a set of characteristics difficult to replicate in mechanical robots. They can be made to move, divide or merge, and they are extremely good at maintaining their integrity, even after a division or when manually manipulated.

Nevertheless, mechanical robots are still the main candidate in order to create artificial life[36], although there has not been any paradigm shift in the information technology field during the last 40 years [37]. The main difference between now and then is that by drastically reducing the size of the transistors the computational power has increased by several orders of magnitude while keeping constant or even reducing the energy consumption. Energy efficiency is one of the main drawbacks of traditional computing because information is transformed and sent back and forth between several different layers of abstraction ([Bérut et al., 2012]). This problem is accentuated in mobile systems, like robots, where energy efficiency and response latency between the interfaces are critical. Behaviour based robots using subsumption or reactive architectures were created to address some of these problems ([Brooks, 1986]), especially the ones which base its foundation in the robots being situated and embodied ([Brooks, 1991], [Pfeifer et al., 2007]). These two concepts are central aspects for all forms of life known, where energy efficiency is crucial for survivability.

An example of life's embodiment is natural selection, because it is a process embodied in its environment since it delegates the evaluation and selection operations into it, easing its computational cost. It has already been described how such processes can be emulated inside computers (Section 1.4.2). The problem is that *in silico* evolution is inherently disembodied, and it is the only human-made one that has been reported as a fully operating system ([Eiben and Smith, 2015]), although open-endedness was never achieved (1.4.5). This leaves the question open about if it is even possible to create an autonomous evolvable system outside a computer.

Eiben *et al.* discussed how to solve this gap between *in silico* evolution and real world evolution using what the authors defined as "embodied artificial evolution" (EAE) ([Eiben and Smith, 2015]). One of the research fields where the authors considered that EAE could be applied is "bottom-up chemo-synthetic systems", which is exactly what was described during this introduction in Sections 1.1.4 and 1.3, where it was explained how oil droplets are viable candidates not only from a prebiotic point of view, but also as a plausible artificial life system.

The authors also described another possibility for the application of EAE, which they named "hybrid mechatronic and biochemical systems". While in "chemo-synthetic system" everything happened *in vivo*, in a hybrid system there would be an *in silico* process where parts of the computation are embodied into the real

[36]It is easier to say "I cannot implement my design because the technology is still not there" than "I cannot implement my design because I do not know how to do it".

[37]The Intel 8086 appeared in 1978. This microprocessor gave rise to the x86 architecture which is still the main one used nowadays. Although it is impossible to set a "breakout" point in the history of computer, apart from the appearance of the first electronic computer in the 50s, we consider the x86 as a paradigm shift from mainstream computers to user oriented ones.

world. Examples of such systems are for example the ones described during Section 1.3.5, where droplets were used as sensors, or going to a completely different research field but within the "hybrid mechatronic and biochemical" models, results like the ones reported by Novellino *et al.*, where the authors connected rat neurons to mobile robots using the neurons as control system ([Novellino et al., 2007]), or like the research reported by Tsuda *et al.* where they used slime mold in a similar way ([Tsuda et al., 2007]). Droplets have been used in similar hybrid systems, acting as "brain", like for example in the work of Suzuki *et al.* where capsule gel robots were self-propelled by oil droplets ([Suzuki et al., 2012]), or the research done by Seah *et al.* in which the authors used oil droplets inside capsules that cleaned an area of oil ([Seah et al., 2013]).

Through this introduction we discussed the origin of life, and in particular the role of protocells during abiogenesis. Protocells are the union of three subsystems: information molecules, metabolism and membrane. In particular, we focussed on droplets as models for such membranes. This thesis focuses on the role of compartmentalization, and how apparently simple compartments could embody a protometabolism and undergo evolutionary processes. The evolutionary processes are studied using evolutionary algorithms, and how they can be implemented using a genetic algorithm. Finally, in order to evolve droplet "protocells", we deemed important the use of automated platforms in order to generate big datasets.

2
Aims

The ultimate objective of our research is to create an autonomous evolvable chemical system. In order to do so we will try to emulate prebiotic processes, because it is known that at that point in time such a system emerged. In particular, we will study the role of compartmentalization, and how simple lipid aggregates could undergo chemical evolution in order to generate advanced membranes that had an active role during the origin of life. It is theorized that such lipid systems are capable of evolution based on the composome model, and we will try to test this model via lab experiments in order to recreate compositional chemical evolution.

In order to do so, the chemistry needs to be aided by two different technologies. On one hand it needs an evolutionary algorithm in order to emulate evolution *in silico* while the evaluation happens *in vivo*. On the other hand, it needs an automated platform that can prepare the experiments, analyse them, and feed back the data. It might seem that by using the aid of these two systems we are *cheating*, but evolution could not happen in a vacuum, but in the context of a full working prebiotic planet Earth, which was already governed by the laws of physics and thermodynamics. Nevertheless, the objective is to minimize as much as possible the role of these platforms. With this objective a "top-down" approach will be followed, where initially a liquid handling robot will be built in order to emulate chemical evolution, and later on this robot will be encapsulated in a simpler platform with the aim of removing some of its dependencies. The main body of work of this thesis is how these two platforms were built, how they were operated, and how the results were analysed. Therefore, the technical aims of this research, reported here, are:

1. Build a liquid handling robot using rapid prototyping tools.
2. Integrate the robot with a genetic algorithm and optimize droplet formulations.
3. Prove that the robot was able to evolve/optimize such droplet formulations.
4. Encapsulate the functionality of such a robot in a single monolithic device.
5. Integrate such a device with an evolutionary algorithm and prove that it can replicate the results from the liquid handling robot.
6. Fabricate different devices with different arenas, and find novel ways of creating different arena topologies.
7. Perform evolutionary experiments with variable environments, proving that the different environments have an impact on the evolutionary dynamics of the system.

In order to test the following hypotheses:

1. From an abiogenesis perspective, we want to explore the concept of chemical evolution applied to membrane compartments. Can these compartments evolve, or were they mere passive elements of abiogenesis?

2. From an artificial life perspective, we want to explore the concept of oil droplets as artificial life entities. Can such simple systems emulate life-like behaviours?

3. From a robotic engineering perspective, we want to explore the use of 3D printers and rapid prototyping tools in order to design and manufacture automated machines that would be able to perform chemical experiments. Can the platforms built using these technologies behave in a similar way to the industrial ones?

3

Dropbot platform

Evolution, once the preserve of biology, has been widely emulated in software, while physically embodied systems that can evolve have been limited to electronic and robotic devices and have never been artificially implemented in populations of physically interacting chemical entities. Herein we present a liquid-handling robot built with the aim of investigating the properties of oil droplets as a function of composition via an automated evolutionary process. The robot makes the droplets by mixing four different compounds in different ratios and placing them in a Petri dish after which they are recorded using a camera and the behaviour of the droplets analysed using image recognition software to give a fitness value. In separate experiments, the fitness function discriminates based on movement, division and vibration over 21 cycles, giving successive fitness increases. Analysis and theoretical modelling of the data yields fitness landscapes analogous to the genotype–phenotype correlations found in biological evolution.

This and the next chapter comprise all the work done for the project known as "Dropbot". The project received this name because its objective was to automate oil-in-water droplet based experiments using a robot. These droplets were based on what is described in the literature as "prebiotic lipid aggregates", and our research focused on the interaction of different surfactant molecules at the interface. The question this project tried to answer was: would formulations of different oils be able to evolve[1] based on macroscopic life-like behaviours similar to movement or division?

In this thesis the Dropbot project is divided into two chapters. The first one, focuses on the platform building process. Its contents are loosely based on the Supplementary Information document [Parrilla-Gutierrez et al., 2014a] made to accompany the main manuscript [Parrilla-Gutierrez et al., 2014b]. The objective of this chapter from an editorial point of view is to take the SI and expand it with unpublished material. The next chapter (Chapter 4) will detail the results obtained using the platform here described.

[1] Or be optimized.

3.1 Introduction

Dropbot is the name given to a robotic platform built as part of this project. The objective of this project was to build a platform that would be able to prepare oil formulations, generate populations of droplets in a Petri dish, record the experiments, analyse the behaviour described by the droplets, and use this analysis to drive an evolutionary algorithm.

The robot itself was built based on a 3D printer because a machine like that provided all the necessary mechanical elements and the only modification needed would be to change the extruder with a liquid system. Therefore, a 3D printer is a machine which is already very similar to our desired final design. We also used one in order to print all the pieces that would automate the different equipment or connect the structural components. The end-goal of this project was not only to build a platform that could generate said experiments, but we also wanted to research how the use of fast prototyping tools, like a 3D printer, could be used to build a highly customized automated platform.

3.1.1 Why a robot?

During the thesis' introduction we discussed similar oil-in-water droplet experiments to the ones we did during this project (see thesis' Introduction p.23, Section 1.3). Something that all of them had in common is that they were performed manually by a researcher using handheld syringes or pipettes. Even we, before the robot was working, had to do them manually. A question to ask then, is "why use a robot?" or even "if all the time you spent building a robot you did the experiments manually instead, you would have finished the project earlier". As we will show during this and next chapter, in total we did around 20,000 experiments, which would render the project impossible without an automated system. Nevertheless, the question "why a robot?" is an important one.

Our ultimate objective is to design an autonomous evolvable chemical system, because that would mean to create life if it is agreed that an evolutionary autonomous system is alive (ie., the chemoton model, see thesis' Introduction p.7, Section 1.1.3). In order for it to be autonomous, there cannot be a robot or any external platform, or especially a researcher. Thus it might seem that building a platform to "create life" defeats the point itself, although life emerged from non-autonomous chemistry, therefore there had to be a "platform", like planet Earth, that guided it. Our strategy is to follow a "top down" approach, where elements of the "evolution helper" are removed in iterative steps. We think that building a robot from scratch which is bound to a particular experiment could be a first step towards a modular system which could be decomposed iteratively.

3.1.2 Prior to Dropbot

Although this project started in October'12, right at the start of the PhD, a similar project was developed just a year before. The objective of that project was very similar to this: to build a liquid handling robot. Its scope, though, was very limited,

mostly constrained by the time we had to do it, and we only focused on generating droplets into a Petri dish and recording the experiments using a camera. It was not mixing oils, or cleaning the dish, or using any algorithm, but it served as inspiration to the robot described in this chapter. In particular, it focused on removing or refuelling droplets dynamically in the Petri dish [Hanczyc et al., 2014]. The frame was built using strut profiles and all the parts were 3D printed. The translation through X and Y was performed using NEMA motors, the syringes automated with a servo motor, and the electronics were controlled using Arduino boards, and using the same firmware as the one used by the 3D printers "RepRap". See Figure 3.1.

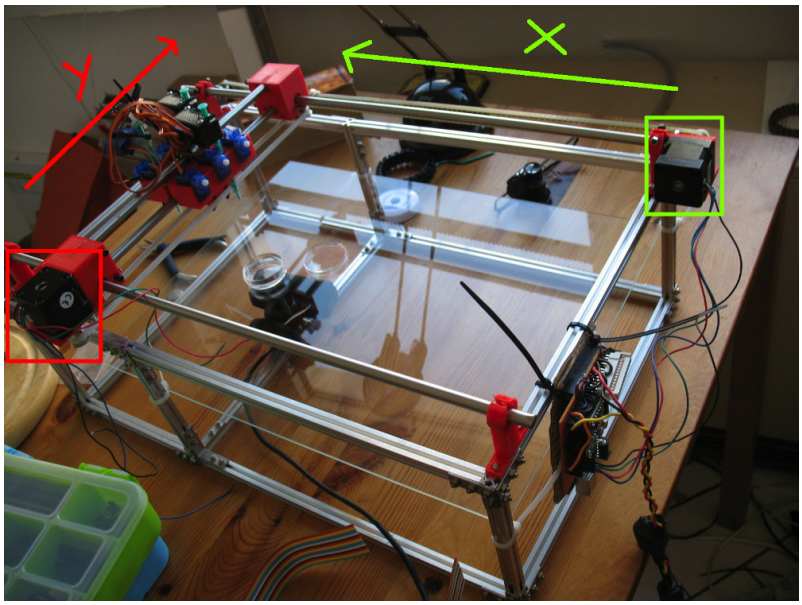


Figure 3.1: Robot built in a previous project, which inspired the one described in this chapter. It also used Arduino based electronics, NEMA stepper motors for X-Y movement, and servo motors to automate syringes.

3.1.3 Project publications

This chapter is directly related to two publications:

- [Parrilla-Gutierrez et al., 2014b] is the main peer reviewed publication. Almost everything discussed in this chapter is explained there. If the reader only cares about the main results, it is recommended to read that publication instead.
- [Parrilla-Gutierrez et al., 2014a] is the "unofficial" supplementary information (SI) of the paper just described.

It can be said that this chapter is an union between these two publications, plus some extra unpublished data. There is a third publication which was partially based on the work developed here, but that it is not discussed in this chapter or thesis because the topic is slightly different:

- [Henson et al., 2015] uses the robot platform here described with very similar chemistry in order to explore its potential use for "unconventional computation". This chapter and thesis focuses instead on origin of life theories as explained during the introduction, thus it was decided to leave this publication aside.

3.1.4 Chapter objectives

- Analyse the use of 3D printers as possible liquid handling robots.
- Automate glass and disposable syringes using motors and rapid prototyping tools.
- Build a liquid handling robot that uses said syringes.
- Integrate liquid pumps into the robot in order to transport in and out big quantities of liquid.
- Write all the software necessary to control the robot and provide a suitable user interface.
- Integrate the robot with an evolutionary algorithm.
- Design a computer vision library to analyse the experiments.
- Integrate the robot with a database software in order to save the experiments.
- Use the robot to generate populations of droplets into a Petri dish.
- Use the robot to clean the Petri dish, so that it can iterate through experiments without human intervention.
- Test the robustness of the platform.

3.2 Building the platform

3.2.1 Hardware

The robot described in this chapter, DropBot, was designed and built based on the RepRap 3D printer platform [Jones et al., 2011]. This platform was chosen as a starting point due to its open-source philosophy, excellent documentation and big user community. These reasons were considered positive as it was expected we would need to heavily modify it in order to fulfil our requirements, and some other popular proprietary projects would make it more complicated.

Early designs of DropBot (see Section 3.1.2 and [Parrilla-Gutiérrez, 2012]) aimed to use a RepRap 3D printer platform directly as a liquid handling robot, replacing the thermoplastic extruder with a liquid handling instrument like a syringe or a pipette. After early prototyping phases, it was decided that this platform was unsuitable and so a new design from scratch was created, re-using modular components from the RepRap project rather than its monolithic design.

This section provides an overview of the methodology applied to the design and implementation of DropBot. Section 3.2.1.1 gives an introduction to the limitations encountered in the direct conversion of RepRap into a liquid-handling robot and the development of the final design used for this project, Section 3.2.1.2 describes the 3D-printed, servo-actuated syringe design that was developed and the liquid handling system used to transport liquid to and from the robot. Finally, Section 3.2.1.3 gives an overview of the electronic system. Figure 4.3 (p.88) shows the final robot prototype, highlighting its main components that will be described during this section.

3.2.1.1 Robot frame

The main limitations found in early prototyping phases, which attempted the direct modification of a RepRap 3D printer into a liquid handling robot, were the small working area and the translation of the Y axis, which in RepRap printers happens by moving the printing bed instead of their extruder tool.

The most common design for a RepRap 3D printer has a working area of 20 x 20 cm, which was not big enough to place all lab equipment[2] required while performing the experiments. Our first prototype had an extended area of 50 x 40 cm.

The second limitation, the movement of stage along the Y-axis, made the design unsuitable for a liquid handling robot, as it could introduce external flows into the liquid phase; essentially, the stage was behaving similarly to a shaker plate. In addition, RepRap's design involved the majority of the Y-actuation mechanism being physically located underneath the stage, occupying space that would be needed for the inclusion of a camera in order to analyse the experiments with a clean field of view.

To correct these limitations, it was decided that the robot tool would be located on an overhead XY-axes, above a static, glass staging area, with a camera located on a separate XY-axes being located underneath this stage[3], viewing the experiments through the glass from the bottom. While this restricted the robot to use only transparent glassware equipment, it was decided that, since this would be the normal mode of operation in any case, this limitation would not impact the final experimentation.

Figure 3.2 shows the final design, used for all experiments of this chapter. The main parts of the design were:

1. The staging area, on which the experimentation lab and glassware were placed.

2. The X-axis, which actuated motion along the long edge of the robot.

3. The Y-axis, which actuated motion along the short edge of the robot.

[2]As it will be described, in our platform the needed equipment was a 96 well plate, a glass Petri dish, and some sort of container for waste.

[3]Although the camera was attached to a XY-axes mechanism it never needed to move during the experiments we developed in this project because we only used one Petri dish. This is why after a while we removed the electronics and the mechanical parts from the camera, and just left it in a fixed position. The electronics and mechanism were removed just to keep the robot as simple as possible.

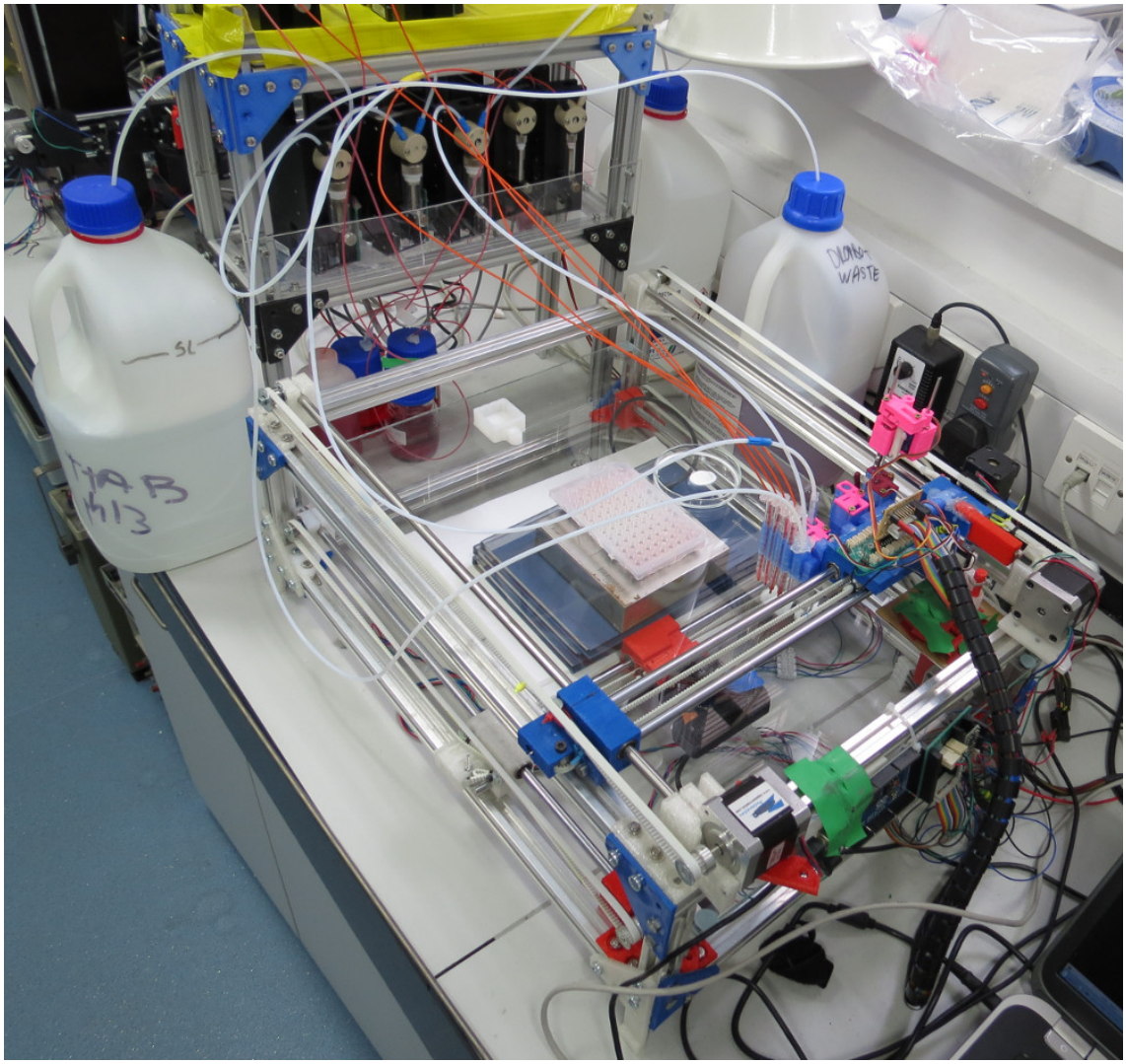


Figure 3.2: Overall picture of the robot. Strut profiles were used to build the frame. They were joined using 3D printed pieces. The carriage which holds the syringes and tubes was also 3D printed. All the mechanisms were based on RepRap 3D printers. The long axis shown here is the X axis, with the shorter being the Y axis. The carriage is shown at home (0, 0) position.

4. The mobile carriange, which contained the manipulation tool, performed formulation mixing, aqueous phase handling, droplet placement and cleaning.

5. The fluid handling platform, which handled the introduction of liquids to the main stage of the robot for further manipulation.

See the experimental chapter (Section 8.1.1, p.176), for the actual implementation of these parts.

3.2.1.2 Liquid handling system

Both servo-actuated syringes and syringe pumps were used to transport liquid phases. The servo-syringes were more precise but were unable to carry as much volume as the syringe pumps. Servo-syringes were used to mix the liquid in the well-plates and to carry small volumes of liquid into the petri dish in order to generate droplets. Syringe pumps were used to transport liquids from source reagent bottles to components on the experiment stage.

3.2.1.2.1 Syringe pumps The main commercial syringe pumps used in the Cronin group are the "Tricontinent C-series". These pumps use two equal stepper motors from the NEMA family: one to actuate the syringe, and one to actuate the valve. Through the years and different research projects, the group accumulated a number of defective commercial syringe pumps, mostly considered inoperative though faults in their logic board. These defective pumps were recycled and used as part of this project. Its original electronics were removed and the motors were connected to custom components (see Section 3.2.1.3).

A total of seven such pumps were used by the robot, mounted on a single platform (see Section 8.1.1.5, p.178). Each pump was equipped with a three-way valve allowing the syringe to be connected through one port to either of the other two. One of these ports was designated as an input and the other as an output. Three of the pumps were equipped with 5 ml syringes: one was used for introducing the aqueous phase (see Section 8.1.4.1, p.182), and the other two for the introducing and removing solvents as part of the cleaning procedure (see Section 8.1.4.2, p.182). The remaining four pumps were equipped with 1 ml syringes; these were used to introduce organic phases, as described in Section 8.1.4.1 (182). All pumps except one were connected via the input port to a reagent bottle and via the output port to the X-Y carriage; the only exception was the syringe pump used for removing the acetone from the petri dish as part of the cleaning cycle.

The tubing set-up at the X-Y carriage can be seen in Figure 3.3. The syringe barrels were used as guides to maintain accurate positioning of the tubing above the working area. Acetone and aqueous phase shared the same barrel, with two independent tubes being fixed inside by hot glue. The support structure was 3D-printed in polylactic acid (PLA) and was attached to the carriage with brass screws.

3.2.1.2.2 Mixing stage The robot used a standard 96-well plate (see Figure 3.4). Each well had its own miniature magnetic stirrer bar and the entire plate was mounted over a stirrer plate. This design was chosen so that multiple experiments

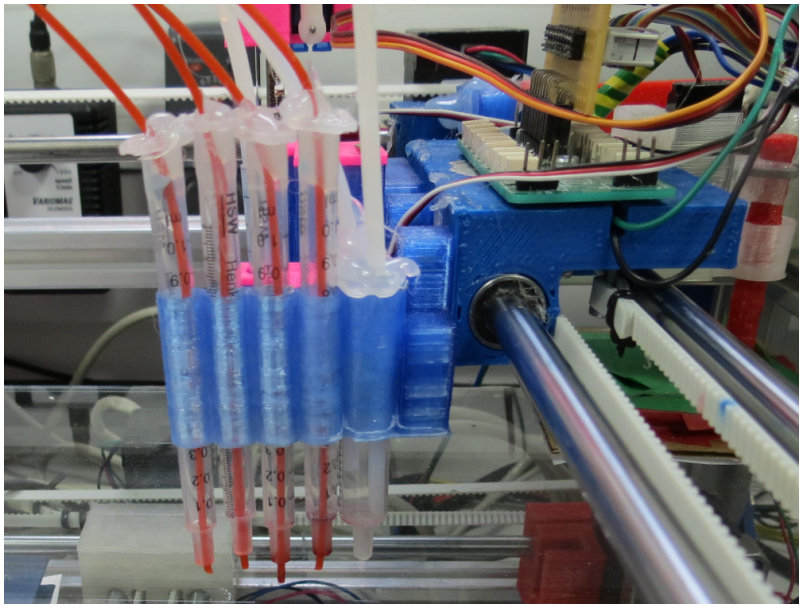


Figure 3.3: Tubing output set-up. Plastic syringe barrels were used to guide the tubes. The first four to the left were the oil phases, and the last one to the right was shared between acetone and aqueous phase.

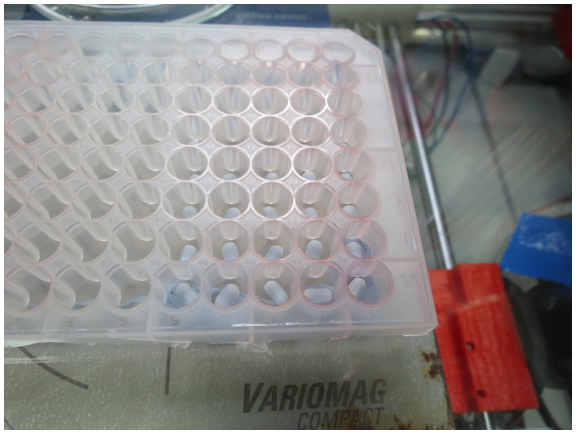


Figure 3.4: "Nunc U96 0.5 ml" well plates, with 'Magnetic stir bar micro PTFE 6 mm x 3 mm" and "Variomag Compact" stirrer plate used the mix the oils.

could be concluded before manual intervention was necessary to clean the mixing area.

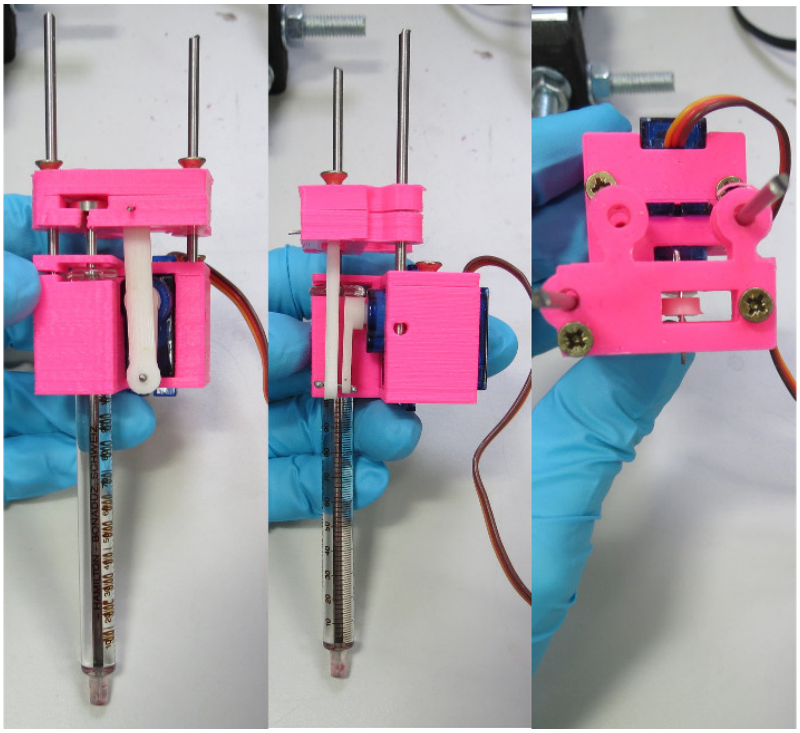


Figure 3.5: Dropbot servo automated glass syringe. The plunger was actuated using a 9g servo motor. Guide rods were used to avoid bending.

3.2.1.2.3 Carriage-mounted syringes As shown in Figure 3.5, an automatic syringe was designed to be used with the X-Y carriage. The casing and structure were 3D printed using PLA. Independent crank mechanisms were used to actuate the plunger of the syringe and to lower and raise the syringe. These consisted of the default servo motor-arm and a 3d-printed piece, joined via steel pins. The crank was aligned with the centre of the syringe itself to avoid unwanted lateral torque. Small-diameter steel rods and teflon linear bearings were used to mitigate rotation away from and towards the servo. The syringes were fitted with a metal needle-tip, as shown in Figure 3.6.

The mechanism to raise and lower the syringe, via a crank, can also be seen on Figure 3.6. On the right of this figure, the same mechanism can be seen to be actuating an unactuated syringe. This syringe had its plunger removed and was instead fitted with a piece of plastic tubing, which connected it to a syringe pump. This syringe was used to remove solvents from the Petri dish during a cleaning cycle. For this reason, the needle taper tip was cut to create a larger absorption diameter. While Dropbot only used one of each syringe type, the robot had the capabilities of using additional syringes, like the empty slot shown on Figure 3.6, or the empty spaces at the other side of the carriage.

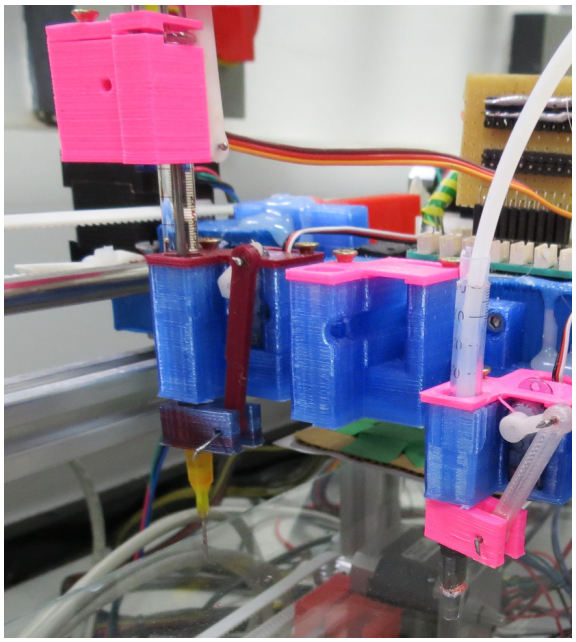


Figure 3.6: Syringe attached to needle and lifting mechanism. The same lifting mechanism was used to raise and lower a plastic needle to remove liquid from the Petri dish.

3.2.1.3 Electronics

The electronics can be divided into three groups. One group controlled the movement of the robot along the X and Y axes, another group controlled the servo motors which actuated the syringes, and a third group controlled the recycled syringe pumps.

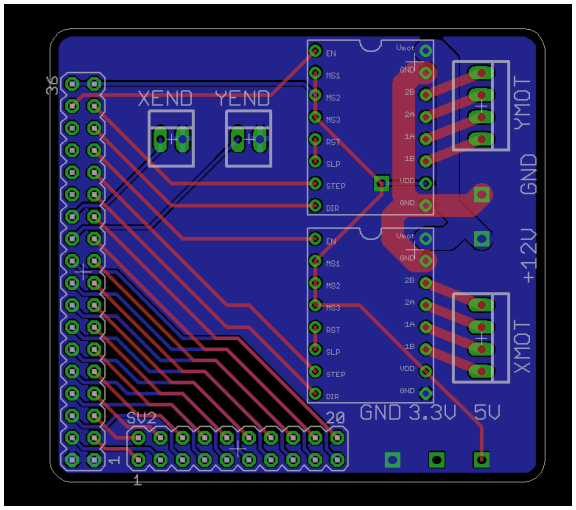


Figure 3.7: Arduino Mega PCB shield with direct control outputs to 2 stepper motors and breakout header to the servo daughter-board.

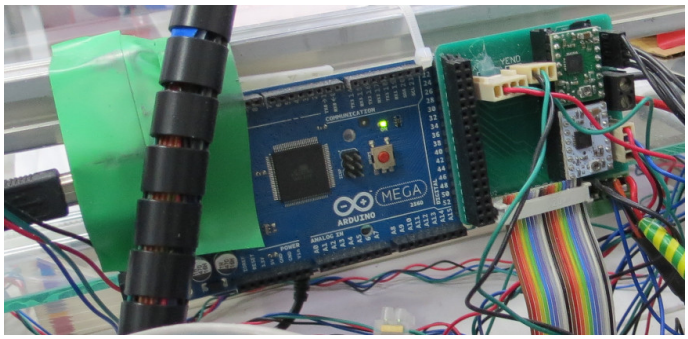


Figure 3.8: Arduino Mega board with the PCB shield described on Figure 3.7.

3.2.1.3.1 Axis control The movement along the X and Y axes was performed using a stripped down version of the electronics used by a RepRap 3D printer, where only the parts required by the X and Y movement were kept: an Arduino Mega board and "A4988 Stepper Motor Driver Carrier[4]".

A PCB shield was designed (see Figure 3.7), which interfaced the Arduino board to the motor drivers and also provided connectors for the motors. In addition, connections were provided for two end-stops, each assigned to a particular axis, for homing purposes. Each stepper driver had 16 pins, but only 3 of them were needed to control the motor: enable, step and direction. Another four pins were connected to the motor itself, and the rest of pins were connected to common high (5V), low (GND) or 12V, as required. The board therefore used eight outputs[5] from the Arduino and interfaced the remainder of Arduino's digital connectors, and a power supply, to a header output for the servo daughter-board (Figure 3.9), because this board was also used to actuate the servos which controlled the syringes. The attachment between the Arduino board and the PCB shield can be seen on Figure 3.8.

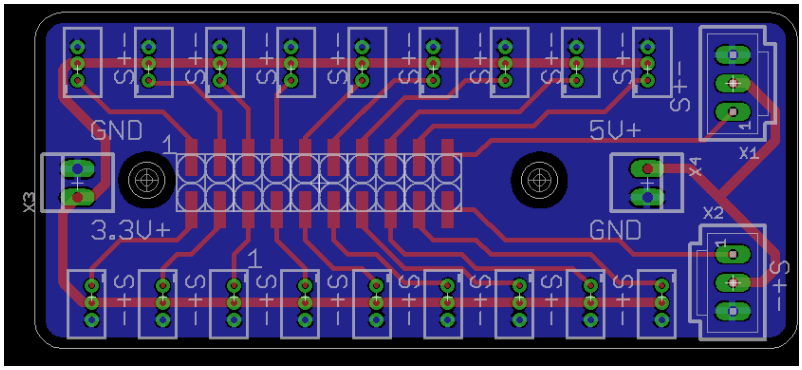


Figure 3.9: PCB designed to map the digital pins from the Arduino board into servo motor sockets. It worked for both 3.3V and 5V servo motors.

[4]Used to power the NEMA family stepper motors which were used for the translation along X and Y axes.

[5]Three for each motor, and one for each end-stop.

3.2.1.3.2 Carriage control The header input to the servo daughter-board (Figure 3.9) consisted of twenty pins, connected to the Arduino via a ribbon cable to the bypass on the XY driver board. The servo board itself was mounted on the X-Y carriage in order to provide both power and control to the syringe-actuation servo motors. Each servo motor took power from a common supply and were connected to a common ground, with the only individual connection being a single PWM data pin to the Arduino. The servo daughter-board was therefore able to service 20 unique servo motors, of which maximum four were used in practice.

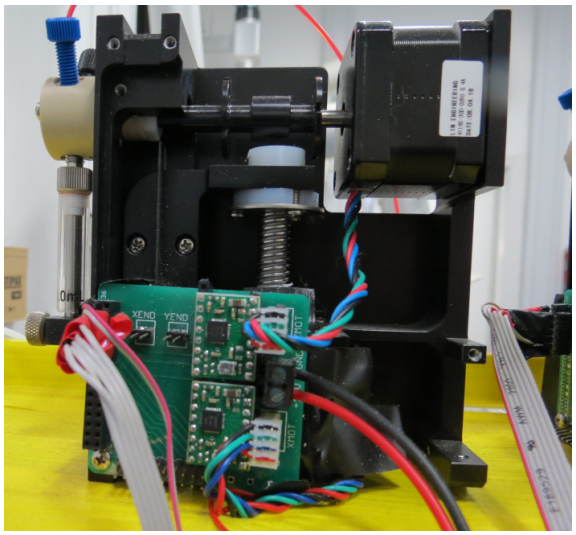


Figure 3.10: Tricontinent pump being used by attaching its motors to the designed PCB shield.

3.2.1.3.3 Syringe pumps control The pumps mentioned in Section 3.2.1.2.1 were faulty only in the function of their logic boards; the mechanical components and motors were fully operational. As their actuation was provided by two stepper motors from the NEMA family, these were therefore compatible with the shield developed for the X-Y axis control (Section 3.2.1.3.1). It was therefore possible to re-use the mechanical components of these syringe pumps by the replacement of their electronics with custom PCBs (Figure 3.10). Since each board controlled two stepper motors, a single board was enough to control each pump. A second Arduino was assigned entirely to pump control, in addition to a second power-supply assigned only to pump motors.

3.2.2 Software

The software components executed from the host computer were programmed entirely in Python, while software components on the Arduino[6] (referred from now on as the firmware) were programmed in C/C++. The software layer was divided into

[6]There were two Arduino boards: one of them controlled the syringe pumps, and the other one controlled the XY axes and the servo automated syringes.

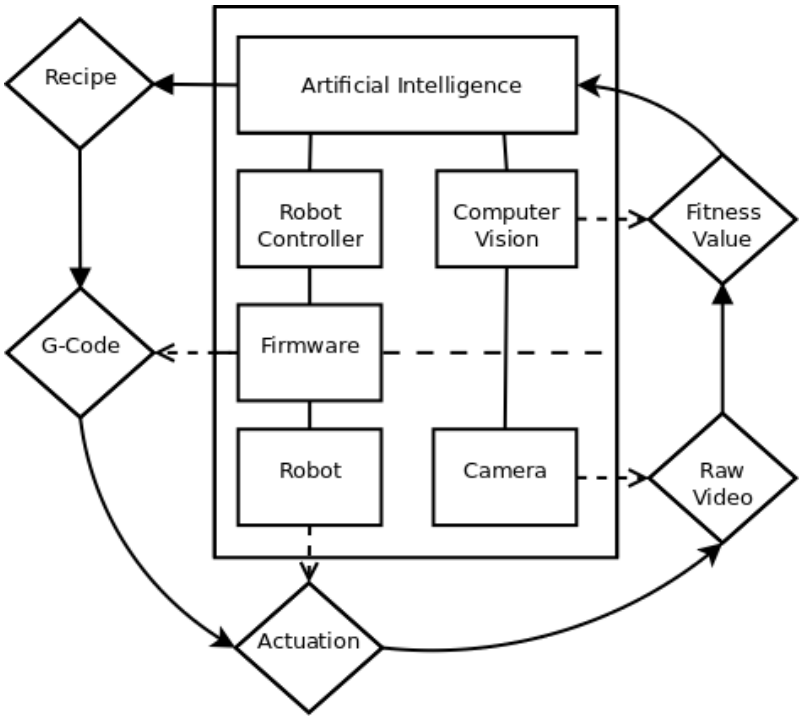


Figure 3.11: Software architecture showing distribution of workflow actors and actuators and their distribution into software packages.

three conceptual modules: *planning*, *acting*, and *sensing*. Figure 3.11 represents the structure of this hierarchy; data-flow occurs in an iterative loop (shown).

The artificial intelligence (AI) component was responsible for coordinating the global experimental plan, selecting droplet formulations based on prior (if any) data and passing these formulations on to the next component. Thus the AI acted globally, with overview of an entire experimental sequence, and the robot controller (RC), firmware (FW) and computer vision (CV) components had control only over single experiments.

The RC took the formulations generated by the AI as input and, acting as an interface between the AI and the physical robot, transformed these numerical recipes into a G-code representation. G-code is a standardized scheduling language ([ISO, 1982]) used in the automation industry to plan mechanical operations (see Section 3.2.2.2 for more details).

The FW consisted of software resident on the Arduino board. It converted the symbolic G-code representation into a sequence of analogue and digital signals sent to the mechanical parts through the Arduino boards.

The CV component interfaced with the camera and converted raw image data, through an algorithmic image processing pipeline, into quantitative and numerical fitness data, which was then returned to the AI.

A multiprocessing software architecture was chosen to make maximum use of time and resources, and to ease development by modularizing the software. Software interprocess communication was used between modules running on the host computer, while USB-serial protocols were used to communicate between the software and firmware. The communication layers between the various components is

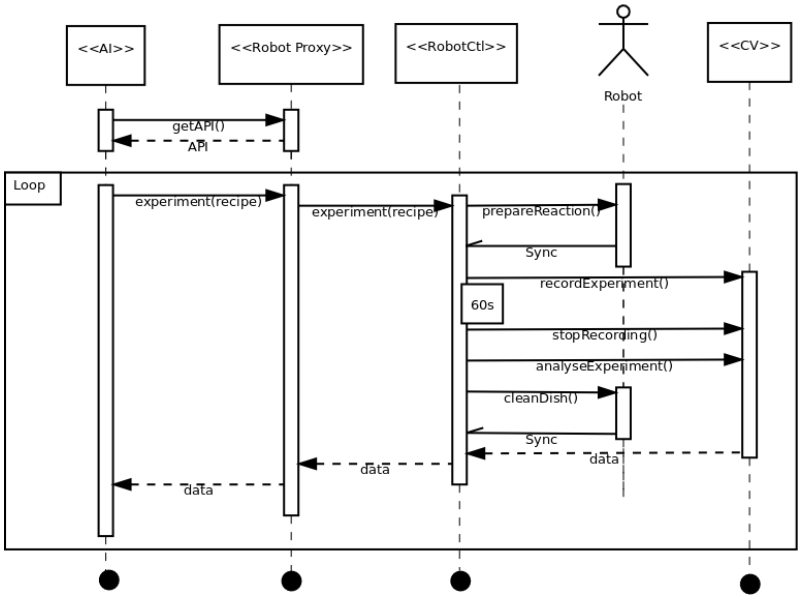


Figure 3.12: UML diagram describing the software work-flow and communication between the different software components.

outlined in figure 3.12. Interested readers should consult the provided source-code for further details.

3.2.2.1 Robot controller

The robot controller aimed to translate experimental procedures form a high-level description into a G-code representation. This function was divided into two steps. The first step was designed to control the robot from a final user point view, and it contained operations like "make droplets" or "clean petri dish". The second step focused on transforming these operations into lower level ones, like "rotate X motor Y steps in Z direction" which later on would be translated directly into G-Code. While technically only the second step was needed, adding a human readable top layer made the development easier. When designing experiments, the researcher only used the first step, being the conversion to G-code transparent to the user.

The core component of robot controller was the Printrun library ([Yanev, 2016a]), and because this library was written in Python by its original developers, all our modifications or extensions were also written in Python. This library was originally developed to send G-code commands from a computer to a 3D printer, usually controlled by an Arduino board, synchronize with it, and listen to instructions that might be sent by the user. The communication between computer and Arduino is performed using a serial protocol through USB. Printrun uses "pySerial" ([Liechti, 2001]) to use this protocol.

In our case, we extended the Printrun library with a new one called "RobotCtl" which described a set of "lab operations" needed to perform an experiment. These operations were of the form "make droplet" or "clean Petri dish". Given a sequence of "lab operations" describing an experiment, "RobotCtl" also translated it to a sequence "low level operations" and then into a sequence of G-Code operations

in order to execute the experiment from a mechanical point of view. The main difference with 3D printing is that "RobotCtl" generated the G-Code dynamically, because at the moment of starting an experiment the exact procedure was not known, since it would be described and refined by the AI over time. We used Printrun's core functionality to send this G-code commands to the robot and synchronize with it.

Using "RobotCtl" the robot could move the syringes to any position around X or Y with a precision of 0.1 mm[7]. Since there were multiple components on the X-Y carriage, the relative distance between these components was hard coded and used to modify the final X-Y position when one specific apparatus was selected[8].

3.2.2.2 Firmware

The firmware layer was directly responsible for the actuation of mechanical parts. The firmware was written specifically for the target Arduino boards, therefore the Arduino development environment was used. The native language of this environment is C++; thus, this was the language used to develop it.

There were two separate Arduino boards, one to control the X-Y carriage (Section 8.1.1.4, p.176) and one to control the fluid platform (Section 8.1.1.5, p.178). As with the RC layer, the carriage firmware was built on top of existent 3D-printer modules. In this case, the firmware core used was the Sprinter package ([Yanev, 2016b]), commonly used with RepRap 3D printers. Functionality with regards to Z-axis movement, temperature control and extrusion of thermoplastic was removed from the base code. On the other hand, functionality was added for control of the syringes (see Section 3.2.1.2.3), via the servo library provided by Arduino ([Arduino, 2016]). The modified Sprinter firmware was therefore able to move along the X-Y axes, able to actuate all servo-motors and synchronize these actuations with the RC component, via the reception and parsing of G-code instructions.

The fluid platform firmware was developed from scratch. Each pump contained two components: A three-way valve and lead-screw actuated plunger. Both of these components used a NEMA stepper motor. These motors were controlled, as with other motors on the robot, via an "A4988 Stepper Motor Driver". A G-code interface was added, with new symbols for pump movement (Table 3.1). The firmware did not check that the parameters sent by the user were valid from a mechanical point of view, thus the RC relied upon coherent operation.

Figure 3.13 outlines the components described.

3.2.3 Computer vision

The computer vision component aimed to evaluate the behaviours exhibited by the droplets for further analysis by the AI component. The robot used a "PS3 EyeToy" to record video data from below the experimental arena. To facilitate visual analysis,

[7]This is the standard precision for the RepRap project. Its main limitant is the backslash from the NEMA motors.

[8]There was a table that related each component to its X and Y. This way, if a user wanted to address any given well from the well plate, it only needed to specify the component, instead of calculating its coordinates

Code	Description
PX	Selects pump X. X ϵ [0...6]
MY	Selects motor Y. Y ϵ [0, 1]
DZ	Selects plunger direction. Z ϵ [0, 1]
SA	Sets speed. A is the number of ms of wait between steps
EB	Sets number of steps. Pumps used are limited to a maximum of 50000 steps

Table 3.1: "PX MY DZ SA EB" Example of G-code operations output by the robot controller.

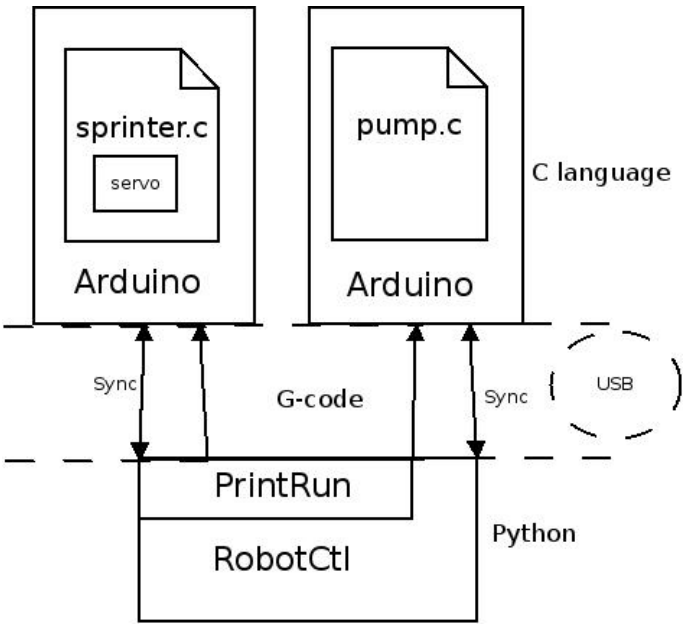


Figure 3.13: Software architecture describing the main components. The robot used two Arduino boards running different firmwares. One (left in this image) extends the 3D printing functionality with liquid handling capabilities. The other (right in this image) controls a set of pumps. The communication with the host-based Python RC was performed using a serial protocol through USB.

a white background panel was used to cover the Petri dish, to block ambient visual noise. A resolution of 640x480 pixels and a frame-rate of 30 FPS was used for video recording. The core component of the computer vision component was the library OpenCV[9] ([Bradski and Kaehler, 2008]), with SciPy ([Jones et al., 2001]) used for some additional analysis. All the analysis were performed using the Python (version 2) bindings for OpenCV.

3.2.3.1 Image processing

The ultimate objective of the image processing component was to fetch a raw frame from the camera (Figure 3.15, top left), identity the droplets present and analyse them to get their position, size, shape and color. Raw frame data was presented to the image processing module in the form of RGB images as received from the camera. No prior information was used to inform the image processing module. Figure 3.14 outlines the complete processing pipeline. As it can be seen in this figure, the image processing module followed two different paths: one focused on using morphological operations, while the other used a background subtraction. Each experiment ran for 30 seconds or 1 minute, and reliable data was needed from the first frame. Foreground subtraction offered the best results when the objects were moving, even at very high speed, while common image processing operations offered the best results when the objects were static or moving at slow speeds, because high speed objects are presented in a frame as a very blurry blob, and any edge detection algorithm we tried failed to detect these shapes.

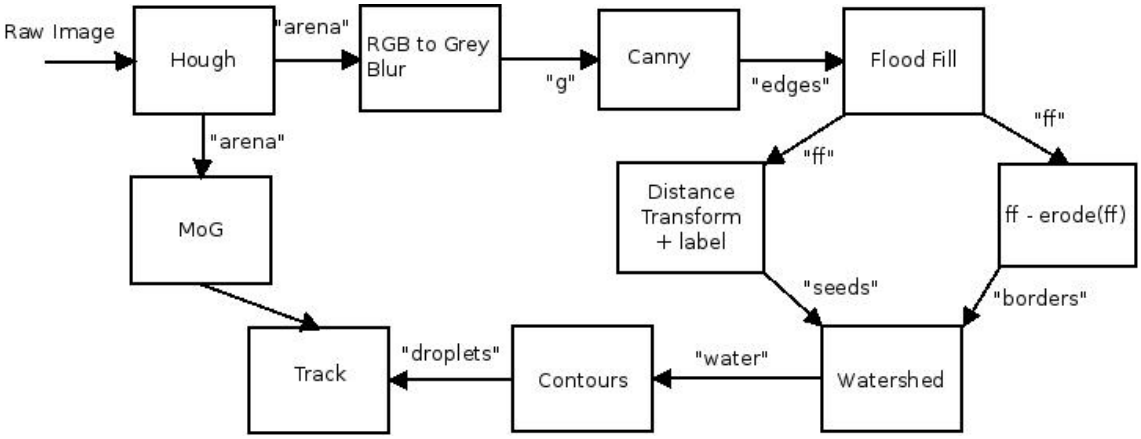


Figure 3.14: Scheme outlining the pipeline of the different techniques and algorithms involved in the droplet detection.

The initial step for both paths began with a Hough transform [Duda and Hart, 1972], used to detect the Petri dish. Once detected, an analytic arena, slightly smaller than the Petri dish, was defined. The aim of this reduction was to remove those droplets that had become attached against to the edge of the dish; these droplets were considered "dead". Only the pixels inside this area were considered, everything else was ignored.

[9]Version 2.4.7

After the Hough transform, the pipeline was split into two parallel tracks. One of these tracks, aimed at long-term analysis, was based on foreground subtraction. The objective of this track was the discovery, and removal, of ambient background from the chemical components of the experiment (droplets). Because a white background was used, only the arena, with transparent aqueous phase, and droplets were present. The foreground subtraction algorithm used a mixture of Gaussians ([Kaewtrakulpong and Bowden, 2001]). The default OpenCV parameters were used for fitting the mixture model, with learning rate set to 0.05. The result of the mixture model was a foreground (droplets) with background information removed. Given that every experiment used the same set-up, and ideal solution would have been to save the background / foreground data between executions, but we did not know how to use OpenCV to do so, and this forced us to start the background subtraction from scratch every time. This meant that during the first seconds the data provided by the background subtraction was very noisy, which forced us to use a very small window of frames to create the model, this had the trade-off that if during the mid or end of the experiment a droplet became static, it would rapidly be assimilated as part of the background. This is why the second track was targeted at analysis of early frames and static droplets, and it was based on a chain of image processing morphological operations.

The objective of the image processing track was the discovery of closed structures, which corresponded to droplets. These structures could have any shape, as long as they could be represented by a connected component. The image processing track began with an RGB-to-grey blurring operation, applied to remove noise. This was followed by the application of the Canny edge detection algorithm[Canny, 1986], resulting in an edge map (Figure 3.15, top right). As contours might miss pixels (as can be seen in Figure 3.15), a dilation operation was applied to fill these gaps (Figure 3.15, bottom left). This was followed by a "flow fill" operation, with the origin at pixel (0,0) (Figure 3.15, bottom right). The result of this operation was the removal of any non-closed structure. The previously calculated Hough transform was then applied, to define the operational arena (Figure 3.16, top left). The next step in the pipeline was the application of a distance transform, to remove noise and to disconnect droplets that may have been artificially connected into one structure as a result of the dilation operation (Figure 3.16, top right). These final connected components were labelled using SciPy; the labels and the borders of the connected components were sent to a watershed algorithm[Meyer, 1992][10], to recover the original size of the droplets (Figure 3.16, bottom left). This concluded the image-processing track of the computer vision module.

The two parallel tracks resulted each in a binary image, where pixels were either 'on' (pixel is part of a droplet) or 'off' (no droplet). These two images were combined, using a boolean "or" operation to give definitive droplet-only images. The contours were then discovered on this combined image (Figure 3.16, bottom right). With this contour map, the area, size, shape, colour and centre of the droplets could be calculated. These data were then passed on to the fitness-specific components of the image-processing pipeline.

[10]This citation refers to the actual implementation used in OpenCV.

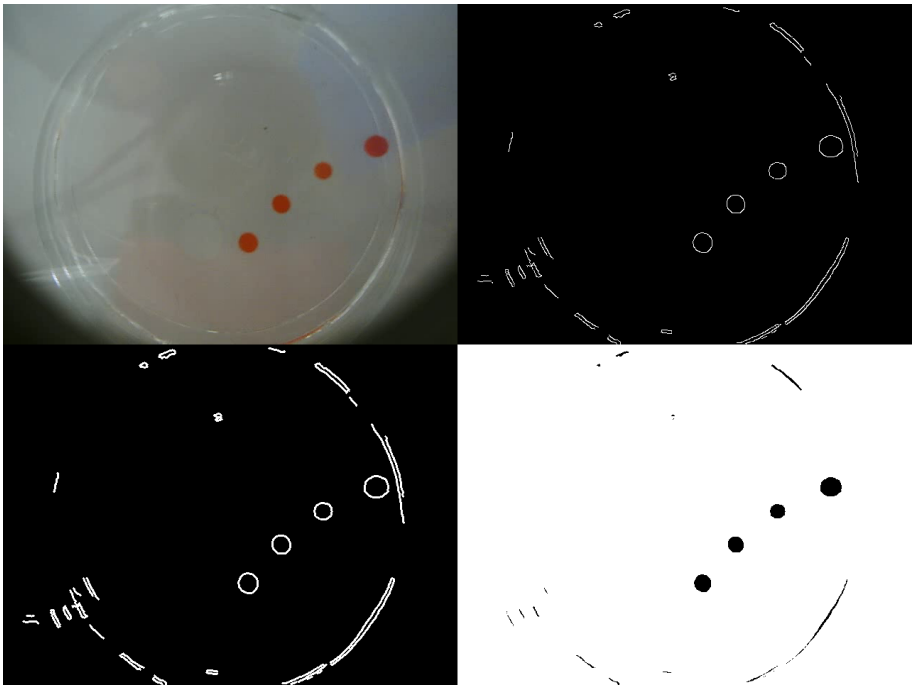


Figure 3.15: Droplet detection, image processing pipeline. **Top left:** Raw frame. **Top Right:** Canny edge detection. **Bottom Left:** Dilate morpho operation. **Bottom Right:** Flood fill operation, seeded at pixel 0,0.

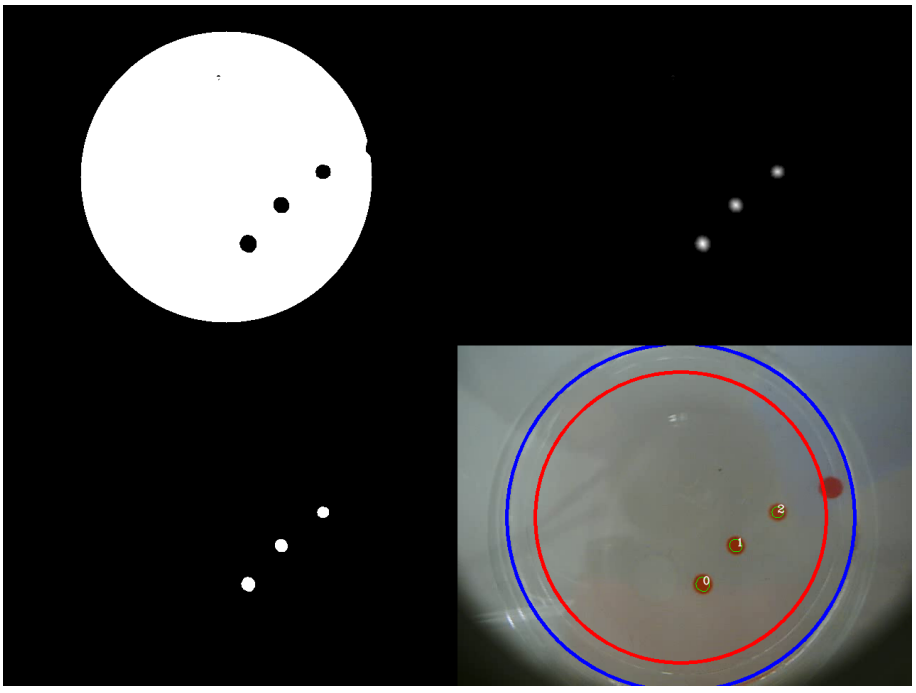


Figure 3.16: Droplet detection, image processing pipeline. **Top left:** Hough transform, dish detection. **Top Right:** Distance transform. **Bottom Left:** Watershed algorithm. **Bottom Right:** Final result. Blue circle marks the dish detection, red circle represents the arena.

3.2.3.2 Blob tracking

Since the camera produced 30 FPS, overlapping consecutive frames to track droplets over time was enough considering the size of a droplet and how much it could move between frames.

Given a droplet d_t in frame t, and a set of droplets D_{t-1} in frame $t-1$, a set of candidates C_{t-1} was built as the droplets in D_{t-1} whose centre was inside an area defined as a circle with 30 pixels of radius around d_t. The best candidate in C_{t-1} was chosen as it was the nearest droplet to d_t using the Euclidean distance. If there were no droplets inside this area, or all the droplets were already assigned to another droplet, this droplet was considered new.

3.2.3.3 Experimental data generation

For each experiment, a data structure describing the positions of the droplets over time was produced. These data were used in order to rate an experiment based on different factors. During this research, the behaviours analysed in this way were "division", "movement" and "directionality". Each of these behaviours represented a "fitness function" when used within a genetic algorithm.

3.2.3.3.1 Division Division was defined as the number of droplets alive at the end of the experiment. We considered as alive any droplet with an area greater than 15 pixels. This fitness function aimed to find droplet recipes that would divide in a controlled fashion, producing viable offspring rather than dividing continuously.

3.2.3.3.2 Movement Given a droplet d, its movement was defined as the Euclidean distance described by its translation between frames t and $t+1$. This translation was described in pixels as the fundamental units of distance.

Since between a pair of frames there can be several droplets moving, the total sum of distances described by all the droplets was divided between the number of droplets, obtaining the average distance translated per droplet. Every experiment ran for 1 minute, containing a few thousand of frames[11]. The average distance per droplet was calculated for every pair of frames, its quantity summed, and then divided by the total number of frames in order to obtain the average translation described per droplet per frame.

The movement-derived fitness is then given by

$$W_{movement} = \frac{1}{MN} \sum_{t=0}^{N} \sum_{i=0}^{M} \sqrt{(x_{i,t} - x_{i,t-1})^2 + (y_{i,t} - y_{i,t-1})^2} \tag{3.1}$$

where N is the total number of frames in the video sequence, M is the total number of droplets observed and where $(x_{i,t}, y_{i,t})$ are the Euclidean coordinates of droplet i on frame t.

In order for a frame to be considered valid, we required that it contained at least two droplets.

[11] Technically with 30 FPS it would be 1800 frames, but the reality is that keeping constant 30 FPS was impossible, so the final number of frames for every video was different, although it was always very near the said 1800.

3.2.3.3.3 Directionality Given a droplet d in frames t, $t+1$ and $t+2$, its position for each frame in the XY plane is denoted by the points A, B and C. Two vectors were defined: $\vec{v}$ as $\vec{AB}$ and $\vec{w}$ as $\vec{BC}$.

By directionality of a droplet it is meant the angle between $\vec{v}$ and $\vec{w}$ which represents the change of direction on a droplet moving pattern. Low values map to droplets which move in straight lines, middle values to droplets which describe curves and high values droplets that vibrate or wobble.

Inverting the dot product formula:

$$\vec{v} \cdot \vec{w} = \|\vec{v}\|\|\vec{w}\| \cos \alpha \tag{3.2}$$

The angle is obtained:

$$\alpha = \arccos\left(\frac{\vec{v} \cdot \vec{w}}{\|\vec{v}\|\|\vec{w}\|}\right) \tag{3.3}$$

For each experiment, the angular rotation, per droplet, per frame was the directional fitness, as given by

$$W_{directionality} = \frac{1}{MN} \sum_{t=0}^{N} \sum_{i=0}^{M} \alpha_i \tag{3.4}$$

3.2.4 Artificial intelligence[12]

For each behaviour to be tested, three Genetic Algorithm (GA) runs were used. Each GA run performed 21 generations, with a fixed population size of 25 individuals and 15 individuals being propagated from the previous generation, for a total of 225 recipes. Each recipe was repeated three times, and the minimum between the mean and the average of these 3 experiments was returned. Each experiment generated four droplets.

A complete test ran the GA 3 times, therefore in total it generated 675 recipes. Each recipe was repeated 3 times, for a total of 2025 experiments. Each experiment generated 4 droplets, for a total of 8100 droplets.

Individuals were fixed length genomes of 4 floating point numbers. The GA used a per-locus probability of mutation. For each locus selected for mutation, a normal-distributed noise function, with a mean of 0 and an standard deviation of 0.1 was additively applied. Each child was always the product of a single crossover recombination between two distinct parents, with the crossover being uniformly distributed along the genome and the same genetic location being used for each parent. Individuals were birthed with parents being selected, without replacement, from the extant pool with probability proportional to the fitness (to the power of some parameter). After birthing and fitness measurement, the population was reduced to a fixed size, with individuals being chosen for death with probability inversely proportional to the fitness.

[12]The artificial intelligence module was developed by Dr. Trevor Hinkley. Although technically it is not part of this thesis, it is an important part of this project, therefore it will be included here for completeness.

The first generation was created randomly. Therefore, because we used four components, our genome length was also four, and each of these components was assigned a random number between zero and one, and then the whole recipe was normalized to sum one. Following generations were built using the roulette selection (Section 1.4.2, p.35). This selection method was chosen as a way of combating the stochasticity of the system, because it aims to spread the genetic pool as much as possible.

Parameter	**Value**
Generations	21
Genome length	4
Population size	25
Carry-overs	15
Per-locus mutation rate	0.3
mutation (SD)	0.1

Table 3.2: Parameters used to generate all GA-derived data presented in this paper.

3.2.5 Side development: Microscope

As explained in Section 3.2.3 the robot was equipped with a standard USB computer camera, and all the experiments performed during this project were done using said camera. We were also interested in using a microscope to study the droplets because with a normal camera it can barely be seen anything happening at the oil-water interface, which is the most interesting part of the droplet system. Initial tests were done using a normal digital microscope available in the lab, and some of the images obtained were very interesting, see Figure 3.17 D - left, therefore we decided to incorporate the microscope into our robot and test if we could obtain similar images for further analysis.

Figure 3.17 A shows the robot microscope assembly, where a 3D printed ring was used to hold the microscope in place. An initial unexpected problem was that a microscope would require its focal plane to be perfectly aligned with the experiment. When using a camera, a few degrees of mismatch are not that important, but with a microscope they are. Our assembly did not take this into account, and it is still a problem to solve.

Figure 3.17 B shows the type of image the microscope was returning. The microscope's field of view did not cover the whole Petri dish, therefore it would need to be able to be moved, which mechanically was possible to do because the camera had the mechanism in place to move around the X and Y axes, although it was never used because it was not needed for our experiments.

Our objective was to always keep the droplet in the centre of the image, and in order to do so, given an image we needed first to detect the droplet, and then calculate its centre. The droplet detection in this case was done using colour segmentation, because we only expected to have a droplet per image, and there was

a clear contrast between the blue background and the red droplets. In particular we used [Otsu, 1979] to threshold the image, see 3.17 B middle picture, and then we used OpenCV's moments function to obtain the centre of the droplet, 3.17 B right picture. The main problem we had was that the lights from the microscope generated false positives, especially when the droplet was against the Petri dish, as can be seen on 3.17 C right image. Nevertheless, we managed to detect a droplet, calculate its centre, and calculate the movement needed from the robot in order to frame the droplet again, although this movement was never tested.

Another major problem we never solved was that considering that the portion of the Petri dish the microscope could see was so small, it was very possible for the droplet to leave it faster than what the microscope could move, or that an experiment started in a scenario where the droplet was already outside the frame. To solve this, on one hand, we would have need to reiterate our algorithm to make sure that it knew when a droplet was inside its viewport, because at the moment it was implemented with the requisite that a droplet would always be present. On the other hand we should develop a strategy the microscope should follow to find a droplet when none are inside its viewport. This was never implemented and the microscope was never fully functional because we tried to do it at the very end of the project, when the publication was already published, and it was decided to move forward with different projects.

It would be very interesting to implement it and study the images obtained with the microscope, because it could potentially offer new fitness functions to study droplet behaviours. As an example, Figure 3.17 D left shows a droplet as seen by a digital lab microscope, not the one used with the robot. This droplet was interesting because it was not rounded: its interface had a strange shape. We did some image analysis trying to guess the number of corners, and the shape of the sides between them, as can be seen on Figure 3.17 D, right. The results were interesting, and we wondered if a similar analysis could be built into the GA, and for example maximize the number of corners or sides with a convex shape.

3.3 Discussion and future work[13]

This platform has been lucky because it had continuation, and some other members of the Cronin Group continued to develop new instruments or improving the whole platform altogether. We could discuss here some of the problems we had, but most of them had already been solved in new platforms. We could also discuss about potential new work, but again, most of it has already been implemented. Therefore, discussing the platform from a technical point of view could be complicated. There are still a few things that are unsolved, and that is what we will discuss here:

- With our current system using commercial syringes and motors, it is almost impossible to generate consistently droplets smaller than 5 µl. The motor can create the pulse, but the droplet won't be released from the needle. There are

[13]Because this chapter only covered the platform itself, a more complete "discussion and future work" section about this project can be found on the next chapter (Section 4.6, p.108) and also during the thesis' "Discussions and future work" chapter (Chapter 7, p.171).

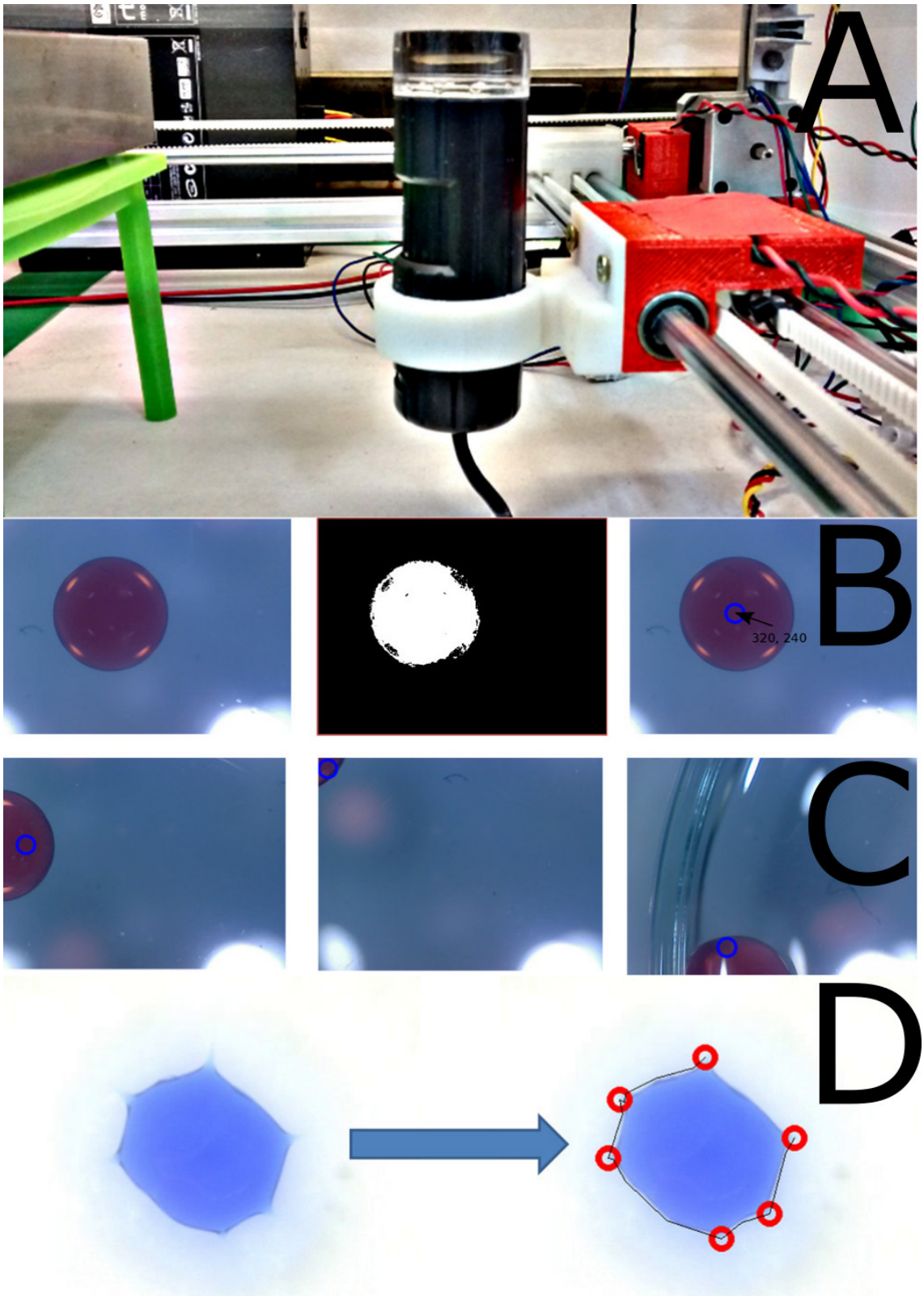


Figure 3.17: Dropbot's microscope. **A:** Robot equipped with the microscope. **B:** Images obtained with said microscope, and centre of droplets detection. **C:** Testing extreme cases of droplet detection. **D: Left** - Image obtained with a digital lab microscope. **Right** - Analysis of the interface shape.

a few possible strategies we have not tried, like making the needle hydrophilic (when generating oil droplets) or to use something like a piezoelectric to generate pulses, instead of a servo or stepper motor.

- The microscope unit as described was never fully functional. It would be very interesting to have it working and analyse its data.
- We did not manage to code new fitness functions for interesting droplet behaviours, like droplet fusion.
- Implement a "droplet catcher". That is, to add a mechanism that would be able to dynamically remove a droplet from an experiment, or to add new elements into said droplet. The concept idea is based on a syringe following a droplet, and "stabbing" it when required.
- Implement active safety components, so that if something goes wrong, the platform would fix it when possible, or stop its function and contact the pertinent research person.

Finally, a few things that are very interesting and that have been implemented in new robots:

- We have designed more robust robot frames with better cable management.
- We have improved our servo actuated syringes.
- A new robot developed by another member of the Cronin group can execute experiments in parallel, meaning that it can execute one experiment every minute, compared to the robot described here, which can execute an experiment every six minutes.

4

Dropbot results

This chapter follows the description of the platform discussed in the previous chapter, and focuses instead on the results obtained.

It will start with a brief introduction (Section 4.1), which aims to complement the one already given in the previous chapter. Then, it will describe chemistry used (Section 4.2), which was already introduced in the previous chapter, and finally the platform itself (Section 4.3).

It will follow with the main results obtained, in particular the ones related to the use of evolutionary algorithms (Section 4.4), and finishing with the discussion (Section 4.5) and future work (Section 4.6).

4.1 Introduction

Biological evolution, a process that relies on populations of replicating individuals who have the ability to inherit, exhibit variation and undergo selection by the environment over successive generations, has been able to create extremely complex and sophisticated biological systems (see [Goul, 1982] or Section 1.4 [p.33] during the thesis' introduction). We therefore hypothesized that the construction of a robot capable of producing well-defined chemical formulations presented as discrete physical objects, in this case oil droplets, could be used as a platform to drive the evolution of populations of the oil droplets. Such a platform would not only require a robot to produce and mix the chemical formulations but also to produce the objects as oil droplets in a dish, and this would be driven by a software programme. The droplets would then be observed by a sensor system, here a video camera, and then the droplet compositions varied using an evolutionary algorithm that could develop the desired properties of the droplets against a well-defined desired test. See Figure 4.1

The reason we are interested in formulating oil-droplet formulations using a robotic system was to see whether we could achieve the robot-driven evolution of these oil droplets, and over multiple successive generations observe correlations between droplet behaviour and composition. Such simple oil droplets could be used as models for primitive protocells since they are characterized by an interface or membrane separating the inside and the outside of the cell; yet they are so primitive that they lack genetic material. However, as we discussed during the introduction, they could still be subject to *chemical evolution* based on their composition (Section 1.4, p.33). This approach therefore paves the way for exploring evolutionary complex chemical systems, as well as a better understanding of the minimal infrastructure

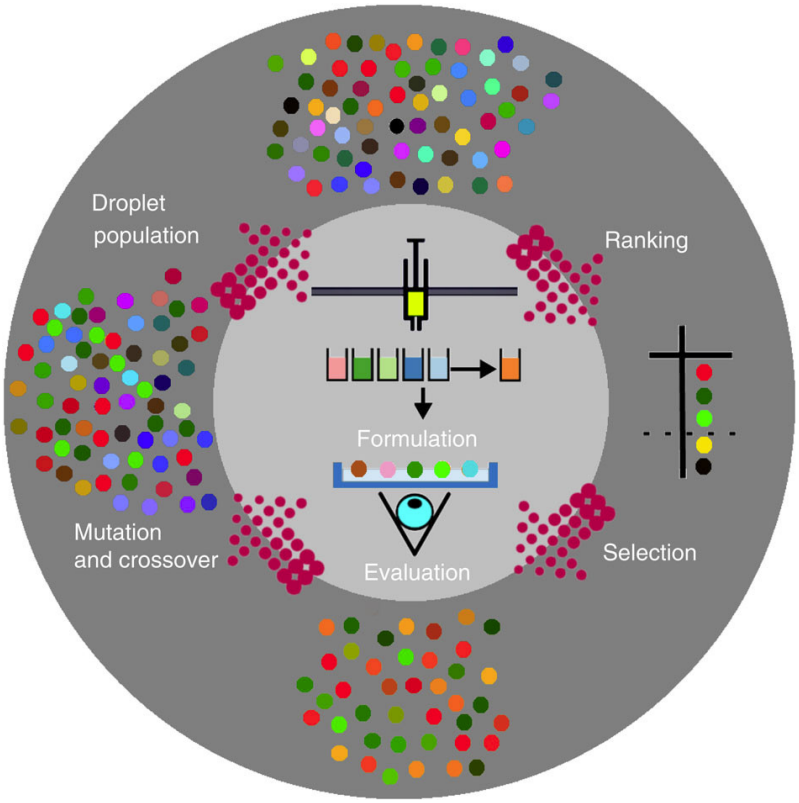


Figure 4.1: The inner circle represents the main part of the robotic process and the outer circle represents the computational algorithm. In the first step a random selection of the droplet formulations is used as the starting "Droplet Population" and this forms the experimental formulations. These droplets are generated in the "Formulation" step and are placed into a Petri dish. The droplet behaviours are then recorded using a camera and then image analysis is conducted against a user-desired property (for example, movement) in the "Evaluation" step. The droplets are ranked in terms of desired property automatically, and the least good rejected in the "Ranking" step allowing a new population to be 'Selected'. Meanwhile, the accepted formulations are used as a basis to create a new 'Droplet Population' after random "Mutation" and "Crossover". This process continues for 21 cycles and the fitness is recorded. Image under CC 4.0 license.

required for the evolution of chemistry outside of biology.

The process of assisting evolution via the robot is itself important since the automation will allow us to do many comparisons, as well as a large number of experiments under well-controlled conditions. In addition, by varying the input "compositions" it is hypothesized that emergent behaviour will result from this coupling. Indeed, the development of hybrid chemorobotic systems capable of combinatorial oil-droplet formation, as well as assisting the process of evolution, has not been possible because of many hardware and software constraints. These requirements include the need to develop a fluid-handling robotic system with autonomous control, sensing the ability to reliably generate a large population of droplets, as well as the automation of the oil-droplet formation – evaluation – cleaning cycle. In addition, solving these issues is also of potential benefit in other areas such as the emerging field of systems chemistry, which seeks to understand and explore the complex chemical systems, networks and emergent behaviours that arise from very simple inputs. In our work we have developed a novel means of exploring complex chemical systems by controlling the interface between robotics, artificial intelligence, complex chemical systems and evolutionary dynamics.

4.1.1 Chapter objectives

While the previous chapter focused on how the platform was built, this chapter will focus instead on the experiments it did and the results obtained.

At the end of the previous chapter the platform was fully working, all the elements were tested individually, and the platform could perform a full cycle where it prepared a formulation, executed an experiment, and cleaned the contents of the Petri dish, leaving it ready to execute a new experiments.

In this project we were not concerned about systematic noise or contamination[1]. Our main worry was the mechanical stability of the platform itself, because it had a lot of mobile parts that would need to execute thousands of experiments without human intervention. This is why the first set of experiments we did was a lattice search, which main objective was to stress test the mechanical robustness of the platform itself.

From that moment on, everything we did were genetic algorithm runs, where the platform tried to optimize the droplet formulations based on the fitness functions discussed previously.

As we will explain, very early on we obtained positive results, thus we showed that our platform was able to evolve droplets based on specific behaviours. From that point on our objective was to obtain fitness landscapes as rich as possible where all the oil components were used.

Finally, because we obtained a lot of data, we also spent a great deal of time trying to find the best way to represent it.

To sum up:

- Test the mechanical robustness of the platform using a lattice search.

[1] As we will be during the next project, see Section 6.1.

- Integrate the platform with a genetic algorithm library and optimize droplet formulations.
- Analyse the results obtained and create a model of the formulation space.
- Prove that the platform was able to evolve droplet formulations.

4.2 Oil droplet system

For our droplet system we chose four components, whose relative quantities formed our compositional genome: 1-octanol, diethyl phthalate (DEP), 1-pentanol and either octanoic acid or dodecane as the fourth ingredient, aiming for droplets that had motility, ability to divide, stability and a range of solubilities, densities and viscosities (see Table 8.1). In this way, the droplets of oil in water provided a chemical compartment with characteristics that depended on the density of the droplet, the interfacial tension between aqueous and organic phases, the polarity of the oil and any reactions that occurred at the interface.

From just a few inputs a wide variety of droplet behaviours emerged, allowing us to explore the potential for the discovery of emergent properties, complex behaviours and the ability to embody evolution within the compositional formulations of the oil-droplet system, see Figure 4.2.

Our initial trial experiments in this chemical space led to the discovery that oil droplets of 1-octanol, when placed in an aqueous surfactant solution at pH 13, underwent rapid movement and binary fission. 1-Pentanol, being less hydrophobic, dissolved in the aqueous phase more readily, resulting in more rapid and less controlled fission. On the other hand, DEP is only slightly soluble in water, resistant to hydrolysis and denser than water, forming well-defined rounded droplets that sank to the bottom of the Petri dish. In contrast, dodecane is less dense than water and very hydrophobic, forming irregular droplets that floated on the surface. Octanoic acid is deprotonated in alkaline media, leading to the formation of an anionic surfactant, able to drive phenomena such as movement or increases in surface area.

Together, these oils occupied a large range of densities, polarities and interfacial tensions, resulting in a parameter space suitable, in size and scope, for exploration as potential protocell models. One of the reasons for choosing formulations is that the chemical stability of the system should result in physical effects rather than chemical reactions.

4.3 Robotic set-up

To explore the dynamics and assist the evolvability of the chosen oil-droplet "protocell" model system, a fully automated liquid-handling robot capable of placing multiple droplets on the surface of a Petri dish of aqueous medium was constructed.

Our robot was designed and built using rapid prototyping techniques and based upon the open source "RepRap" 3D printer, where the extruder was replaced by a liquid-handling carriage and the software adapted for these new capabilities. This resulted in a fully automated, highly customisable robotic platform with four input

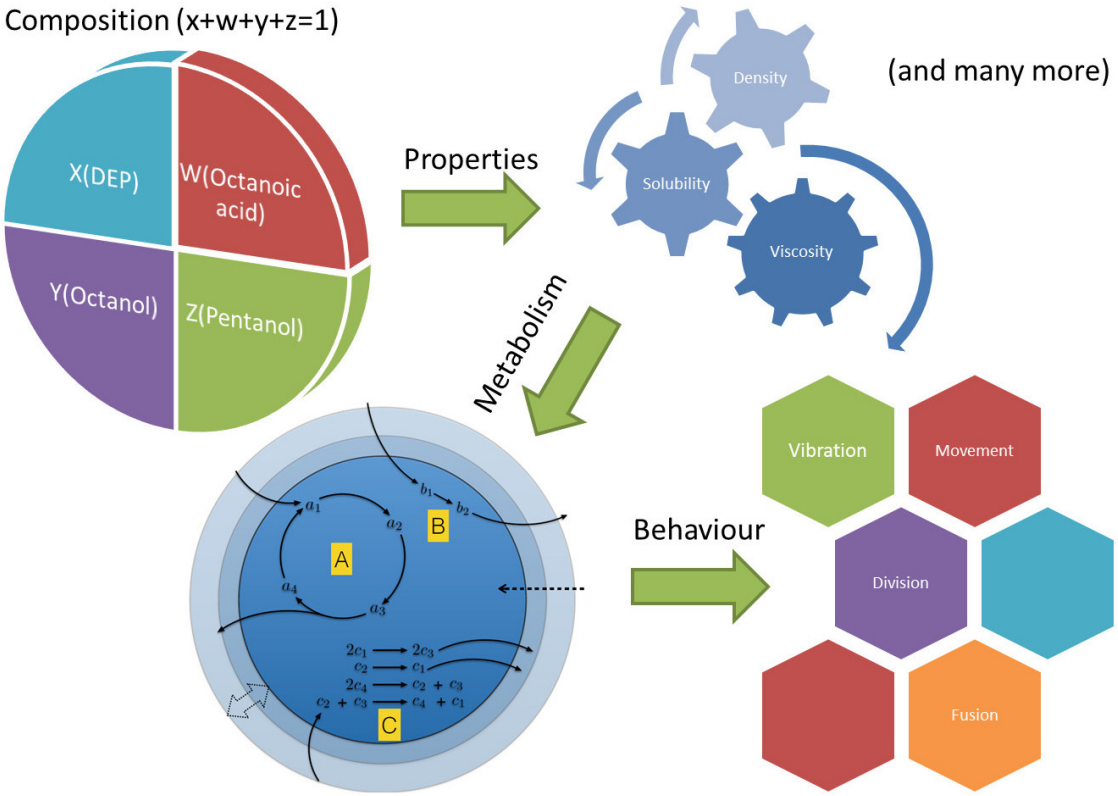


Figure 4.2: Each of the droplets contained a different composition based on a given formulation. This composition had a direct impact on some of its chemical properties, like density or solubility, which at the same time had an impact on the droplet "metabolism", which refers to the flows that went in or out the semi permeable interface. Ultimately, these flows had an impact on the droplet macroscopical behaviour. During our research we focused on the first and last step, and the chemical properties and droplet metabolism were transparent to us. Droplet metabolism image is from [Shirt-Ediss et al., 2015] CC BY 4.0.

chemicals, a series of pumps, a well-plate array for mixing of the chemicals and an array of syringes on a robotic stage, see Figure 4.3. Further, the modular design means that the robot could be easily configured for a variety of different chemical, material or formulation-based studies.

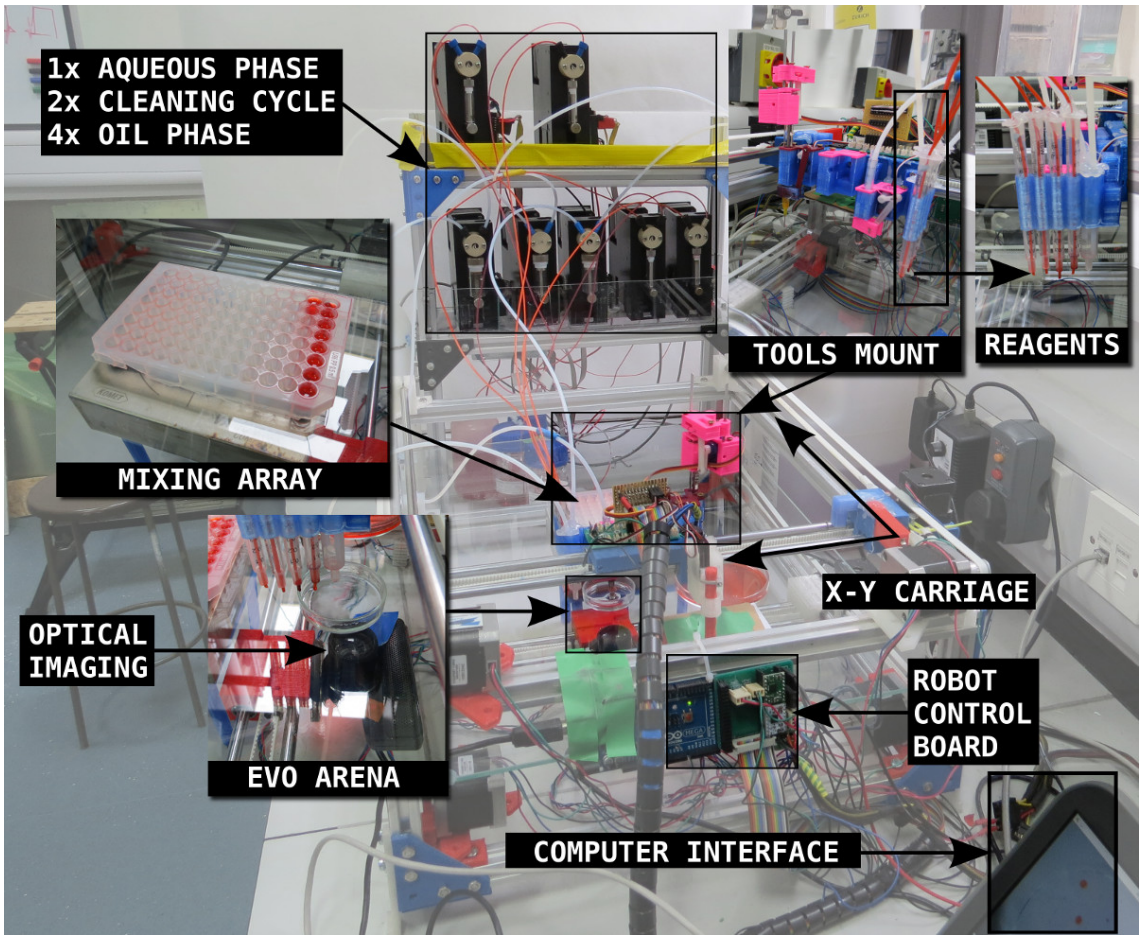


Figure 4.3: Dropbot main prototype used during this research. The syringe pumps shown at the top of the picture were used to handle big quantities of liquid, while the tools shown to its right were used to handle smaller quantities, like droplets. The reagents were mixed in a well plate, and the experiments performed inside a Petri dish, and recorded using a camera. All the components were controlled using Arduino boards, which at the same time were controlled by a computer.

In our experiments, four 5 µl droplets were placed on top of the aqueous phase and a video of the resulting droplet behaviour was recorded from beneath the dish using a camera at a resolution of 640 by 480 pixels at 60 FPS. The X-Y carriage of the robot carried four nozzles, which were connected to syringe pumps to produce the reagent mixtures for the droplets in a single well of a 96 well plate. Each well was stirred with a magnetic stirrer bar to prepare the samples, before being withdrawn by an automated syringe. The formulations were then transferred to produce populations by releasing the droplets from the syringe into the aqueous phase (20 mM aqueous TTAB pH 13.00). A fifth nozzle was used to add the aqueous phase into a Petri dish that acted as the experimental arena. A cleaning nozzle was also present on the

carriage, allowing for automatic cleaning. After the experiment, the entire contents of the dish were automatically removed and the dish was cleaned with three washes of acetone followed by three washes of aqueous phase. During the acetone wash, the syringe was also cleaned with acetone.

4.4 Experimental results

4.4.1 Lattice search

Once the robotic system as described was fully assembled and operative, we conducted a series of individual experiments to test its mechanical capabilities, like the resolution at which it could prepare the oil mixtures, which was around 10 µl, and the smallest droplets it could generate consistently, which were around 5 µl.[2]

The next step was to test its mechanical robustness in a continuous experimentation mode, and that is why an evenly spaced combinatorial "lattice search" was conducted over the four-chemical search space, whereby the chemical formulations were composed as 8-bit entities, with 2 bits assigned to each of the relative quantities of dodecane, pentanol, octanol and DEP. Under this regime, a total of 225 unique combinations were possible (NB: as, for example, 1,1,1,1 is exactly the same as 2,2,2,2 and 0,0,0,0 is not a valid formula). After mixing, the formulations were placed into a Petri dish containing the aqueous phase to form droplets, and a 1-min video was captured. Each formulation was tested twice.

Not only did the robot was able to perform the full lattice search without any mechanical failure[3], it also discovered and characterized a total of nine distinct behaviours, as described in Figure 4.4, displaying a great deal of unexpected different droplet physical characteristics.

One of the most interesting behaviours discovered using the lattice search is displayed in Figure 4.5, where it shows droplets that would divide and fuse exactly once. Although each recipe of the lattice search was repeated twice, we decided to repeat this behaviour several times in order to test if it could be visually reproducible, and the results were satisfactory because based on our own visual assessment the droplets represented a similar behaviour each time the experiment was repeated. As far as we know there is not any automatic fusion-fission cycle reported in the literature. We never fully explored this behaviour, but its underlying mechanism is very interesting because it is similar to many biological processes where cells or molecules go through similar cycles.

To better visually represent the behaviours shown by the droplets during the lattice search, a self-organizing map analysis (SOM) [Kohonen, 1982] was built using

[2]The automated syringes could displace volumes lower than 5 µl, and this mostly depended on the total volume of the syringe. What was more difficult was to generate droplets, and in our experience droplets smaller than 5 µl on average used to stick to the tip of the needle and never be released.

[3]The robot described here was operative for around a year, and only once it failed mechanically. In particular, the mechanism that lifted or lowered the whole syringe became disconnected because the crank shaft broke. This meant that the needle of the syringe went all the way down to the glass plate, pushing everything while moving around. This problem happened way after the lattice search was conducted, near the end of the experiments described in this chapter.

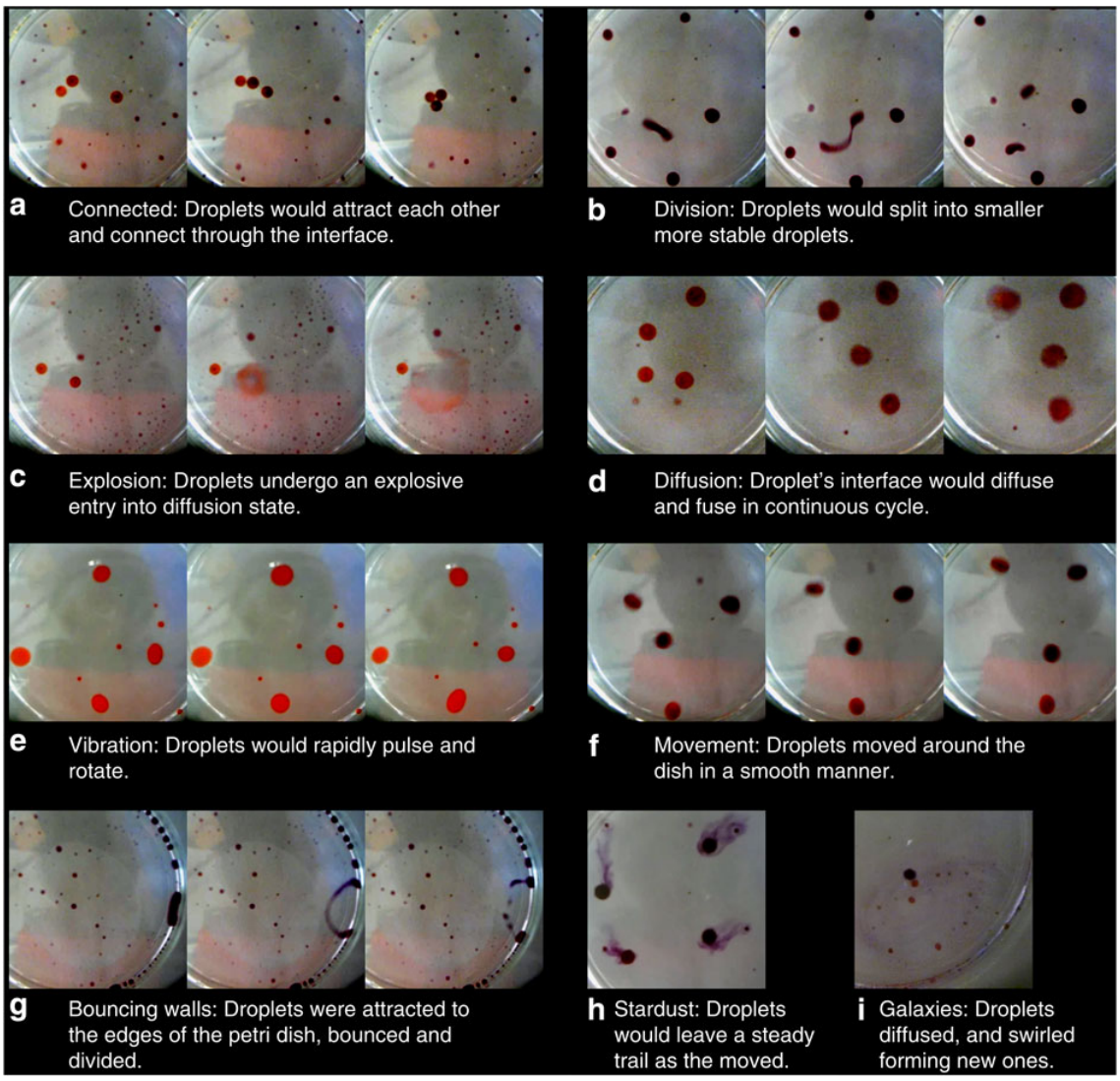


Figure 4.4: Behaviours discovered during the lattice search. Photographs of the droplet behaviour as a function of time (from left to right) for all the traits (given in a–i) except the 'stardust' and 'galaxies' where just one image is shown. Image under CC license.

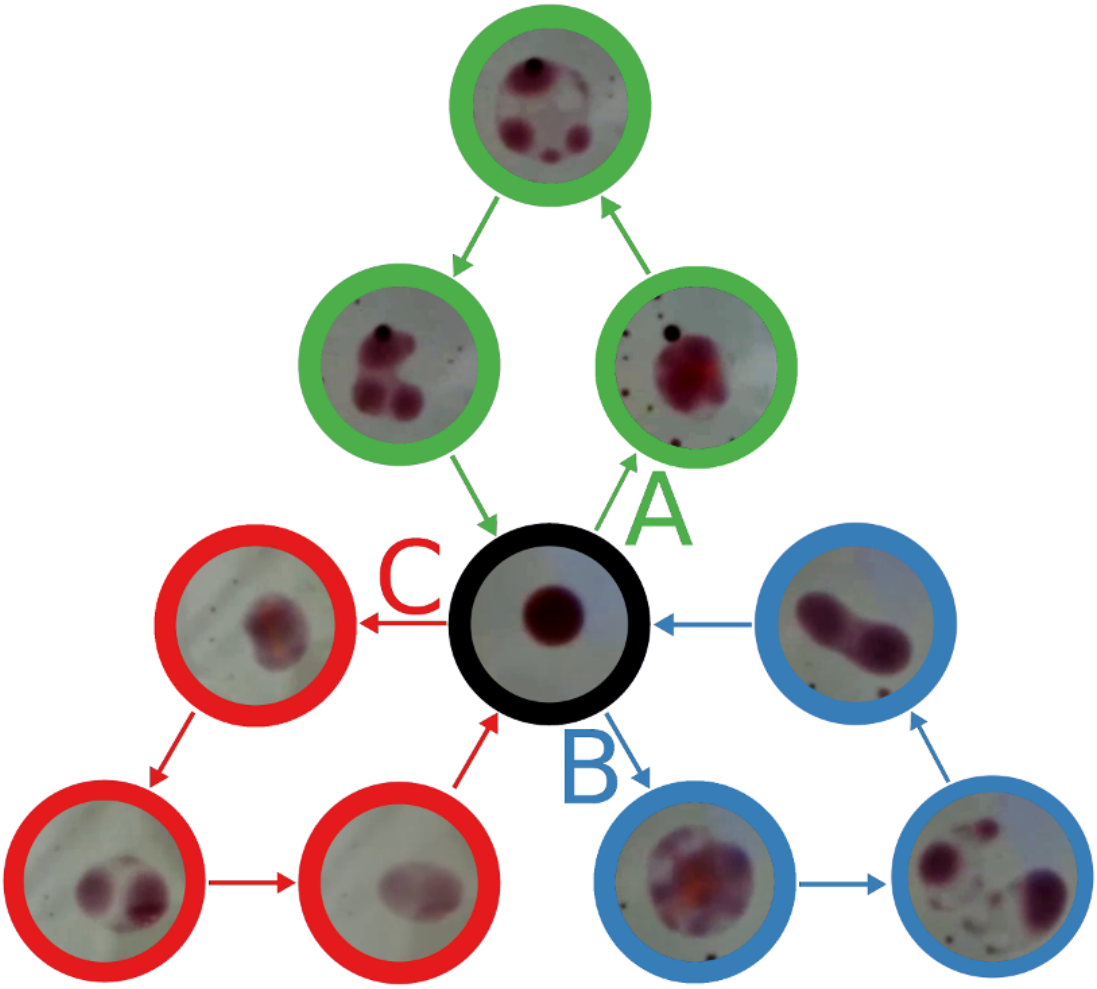


Figure 4.5: Droplet fusion-fission cycle. These droplets would divide and then fuse exactly once. This figure shows three different repetitions of this process, in three different experiments. Following the arrows and starting from the centre, in each case the first step represents when the droplet started to diffuse, the following step represents when the droplet divided, and the last step represented when the droplet fused again.

the generated data. SOMs are a unsupervised learning method used to cluster data. To generate the training data, each of the 225 unique compositions was manually assigned a behaviour from the ones described on Figure 4.4. This assignment was done based on visual assessment by one of the researchers. Each circle in Figure 4.6 represented a "node": the fundamental unit of output from the SOM. A node does not represent a composition, but several of them. In this case, because there are 255 compositions, but only 100 nodes, we can assume that each node will represent between two and three compositions. Because SOMs are unsupervised methods, the network of nodes and its clustering was done only using the recipes, while a colour identifying a specific behaviour was assigned on a second pass based on the previous manual assignment.

The objective of this analyses was to identify any possible correlation between formulation and behaviour, and it can be seen that chemical composition varied across both the X and Y axes in a significant ways, which was not as obvious when watching the real experiments. As said, each behaviour was assigned an individual colour and it can be seen that behaviours clustered together in space, and therefore in composition. Each cluster of behaviours thus represents a phenotypic "island" within the composition/behaviour mapping.

The chemical system described was very sensitive to the physical properties of the droplet, like aqueous solubility, surface tension, density and viscosity. Although we do not have concrete data to explain the chemical mechanism that drove these behaviours[4], we think that the driving force was related to the balance between the internal forces within the droplets driven by instabilities such as the Marangoni instability[Scriven and Sternling, 1960], by the phase separation of the different organic components and by their residual solubility.

4.4.2 Baseline experiments

Before showing the results obtained during our evolutionary experiments, we will focus instead on purely random experimentation that we will consider our baseline. We expect that the use of algorithms will improve this baseline.

In order to do so, we generated 20 generations of 50 individuals each, and each individual's composition was always generated randomly. Thus, this baseline marks the threshold that is achieved by pure random experimentation, and if our algorithms are better than it, they should show better results because we expect the fitness values to improve through generations. The fitness function used was "movement", as described before, and the results can be seen in Figure 4.7.

The baseline experiment was repeated three times, and in each case it can be seen than no growth was achieved, and the average fitness per generation seems to go up or down in a random way, as expected.

This data was collected by Dr Jonathan Grizou using another platform called "DropFactory", which is a similar robot in functionality to the one described here. The chemistry used during these experiments is exactly the same as the one described during Section 4.4.3.3.

[4]We are studying this currently, but it is unrelated to the work done in this thesis.

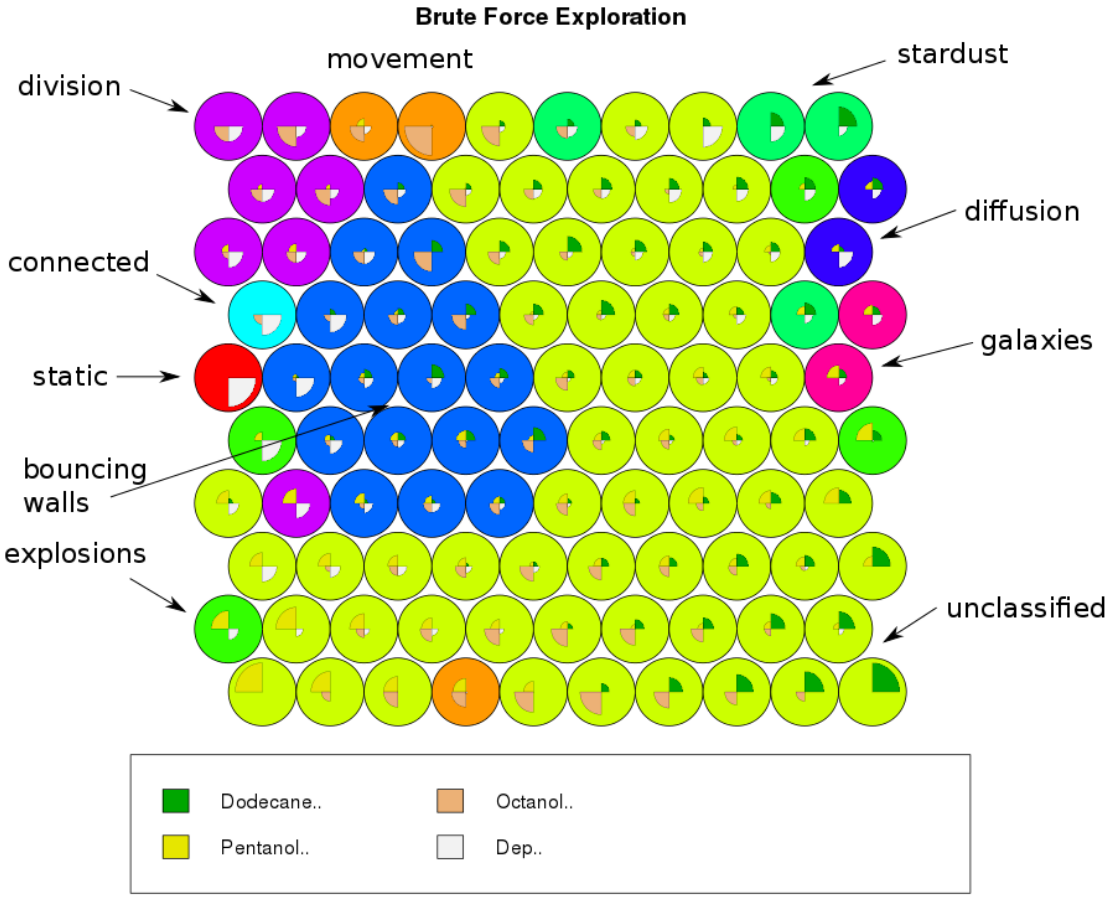


Figure 4.6: Self-Organizing Map of droplet behaviours across the formulation space. Each circle represents a resultant node as generated by the algorithm, with colours assigned to individual behaviours on a second pass using the information provided manually by the researcher. Chemical composition varies according to spatial coordinate with neighbouring coordinates having similar composition. The clusters tagged as division, movement, stardust, diffusion, galaxies, explosions, and bouncing walls; represent the behaviours described on Figure 4.4. The behaviour here described as static was related to droplets that did not move. The behaviour here described as unclassified is related to formulations that did not produce droplets.

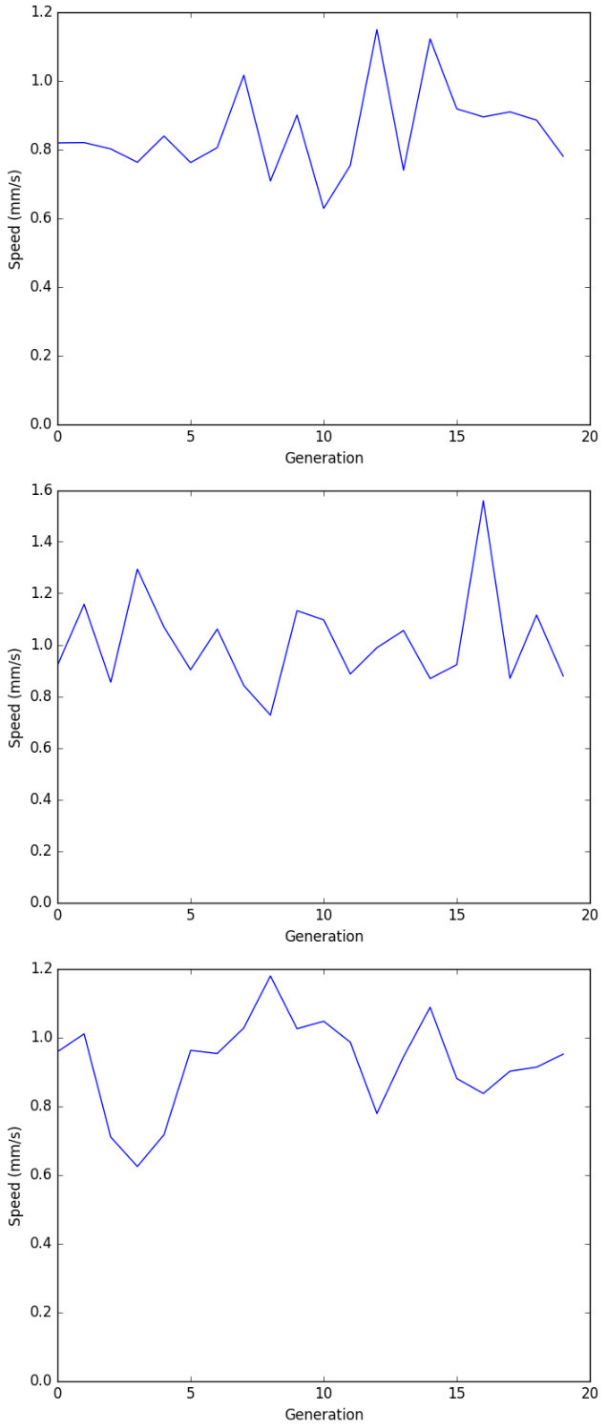


Figure 4.7: Baseline experiments. Twenty generations were performed, each of them containing 50 individuals, and each of these individuals was always generated randomly. The fitness function used was "movement", and for each generation its average value is plotted here.

4.4.3 Evolutionary experiments

To explore the possibility of evolving oil droplets as protocell models for different traits, behaviours or fitnesses, a series of experiments using an evolutionary algorithm with our robotic platform were performed. The algorithm was set up with point mutation and two-parent recombination, using roulette selection, both to select parents for each subsequent generation and to remove "dead" individuals. Each set of inputs consisted of four real numbers, which were constrained to sum one, and which scaled with an 8-bit resolution. Each of these numbers directly corresponded to the relative quantity of one of the substances used in the formulations for each optimization.

4.4.3.1 Using dodecane

Our first set of experiments used as inputs 1-octanol, 1-pentanol, DEP and dodecane. The GA parameters were as described, and the fitness functions tested were division and movement. Division was defined as described, but movement in this case was slightly different, because we measured the total distance in pixels for all the droplets during the whole video. We realised that this would reward formulations that would divide and move, because the higher number of droplets the higher the total movement would be, and that is why in the next iterations we used distance divided by number of droplets, as explained before, but not in this case. In total we did GA three runs for division and GA one run for movement.

The results can be seen on Figure 4.8. They were good because it was the first GA run we did with the whole robotic system working, and not only was the platform stable, but we could see that in every case the fitness values increased through generations, meaning that the robot was able to optimize the formulations. When checking the average genome per generation, we could also see that the surface plot had a defined profile at the end of the experiment, which is a good indicative that the fitness trajectory is navigating through the fitness landscape, especially when compared to the surface profile at generation 0, which is almost flat because that generation was generated randomly. Here we are only showing the genome evolution for one of the GA runs in the case of the division fitness function where the final formulations optimized for DEP and 1-pentanol. In the other two GA runs the genome evolution was more similar to movement, where DEP and 1-octanol were the components with more presence. These results were interesting because it meant that the fitness function could be optimised at two different points in the genome landscape.

On the other hand, it can be seen than in both cases dodecane was almost removed from the system, and it probably would have been if we had let the system run for more generations. This was not an ideal scenario for our experiments, because we were only using four inputs, and if one of them was removed so easily, it meant that our genome landscape only had three components. Dodecane was an important actor in some of the behaviours shown in Section 4.4.1, but in the case of the behaviours we were trying to optimise here it had clearly a negative effect. This is why we decided to replace it with octanoic acid. The reason for choosing octanoic acid was because we thought that one of the main reasons dodecane was removed

from the system was because it was not a surfactant, therefore its presence was not being part of anything happening in the interface. Octanoic acid is a surfactant molecule, and more interestingly it has an OH group which will be deprotonated at the very high pH of our aqueous phase. Thus, it was a very interesting molecule to test in our system.

4.4.3.2 Using octanoic acid

Figure 4.9 shows the results of the GA runs performed using octanoic acid, 1-pentanol, 1-octanol and DEP; for both the division and movement fitness functions, which in this case was already the updated function that measured the movement per frame divided by the total number of droplets.

The results were interesting, but when using octanoic acid a new problem appeared: the fitness landscapes were very flat because there were not big variation of recipes through generations. No matter what the formulation was, its fitness value and visual behaviour was always very similar.

In the case of division, for example, we can see on Figure 4.9 that after 20 generations the number of droplets at the end of the experiment was four, meaning that there was no division at all, while after 30 generations it was near six, when before using dodecane it went to around ten after 20 generations. It can also be seen that the genome evolutionary landscape through generations was very flat, with almost no difference until generation 20.

In the case of the movement fitness function, from generation seven onwards it made almost no improvement, and again the genome evolutionary landscape stopped changing.

In both cases, as happened with dodecane, octanic acid seemed to be removed from the system in order to optimize the formulations, which was exactly what we were trying to avoid. On the other hand, when using octanoic acid we detected a new behaviour that we could easily optimize: the droplets would move in a vibrating pattern.

Based on these results we decided to move forward with two variations: (a) instead of using pure octanoic acid we would dilute it with the other components in 20-80 ratio (20% octanoic acid), this way we expected the fitness landscapes to be more interesting with higher variations of recipes between generations, and (b) we would add in our system vibration as fitness function, with the objective of optimising for three different fitness functions that would use the four components of our system.

4.4.3.3 Using diluted octanoic acid

The system described as follows is the one used for the rest of the experiments, and was used to generate the data that appeared in the published article. The four inputs chosen were DEP, 1-octanol, 1-pentanol and octanoic acid diluted with each of the previous components in a 20 to 80 ratio as described in Section 8.1.4.1.

For each of the fitness functions three repetitions were done, one where octanoic acid was diluted with 1-octanol, another one where it was diluted with 1-pentanol, and a final one where it was diluted with DEP. We did not observe any difference

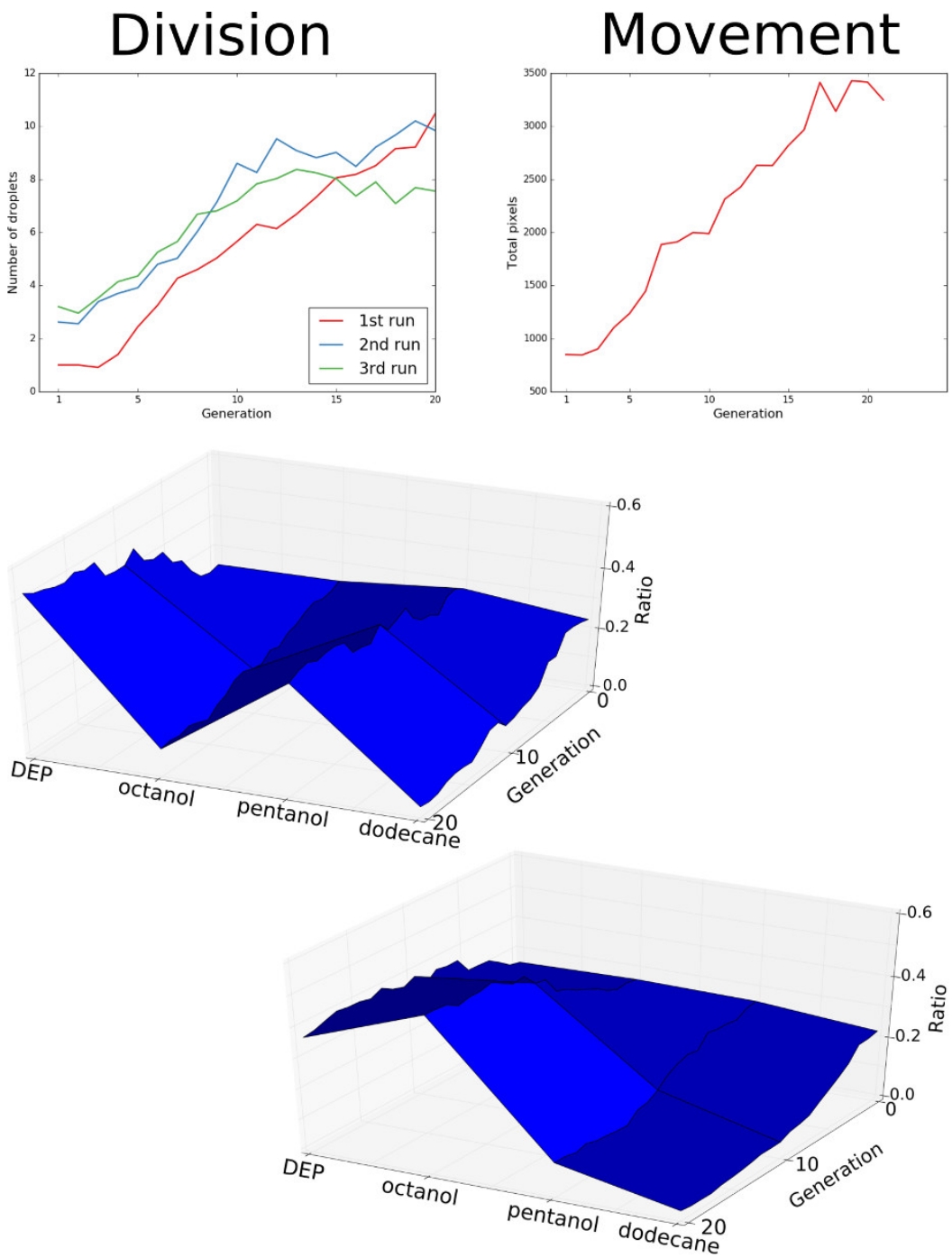


Figure 4.8: GA runs using dodecane. The fitness functions were division (number of droplets at the end of the experiment) and movement, which in this case counted the total number of pixels moved by all droplets. The "division" linear plot relates to the middle left surface, while the "movement" linear plot relates to the bottom right surface. The "ratio" label in the surface plots' Y axis represents the average ratio within a generation.

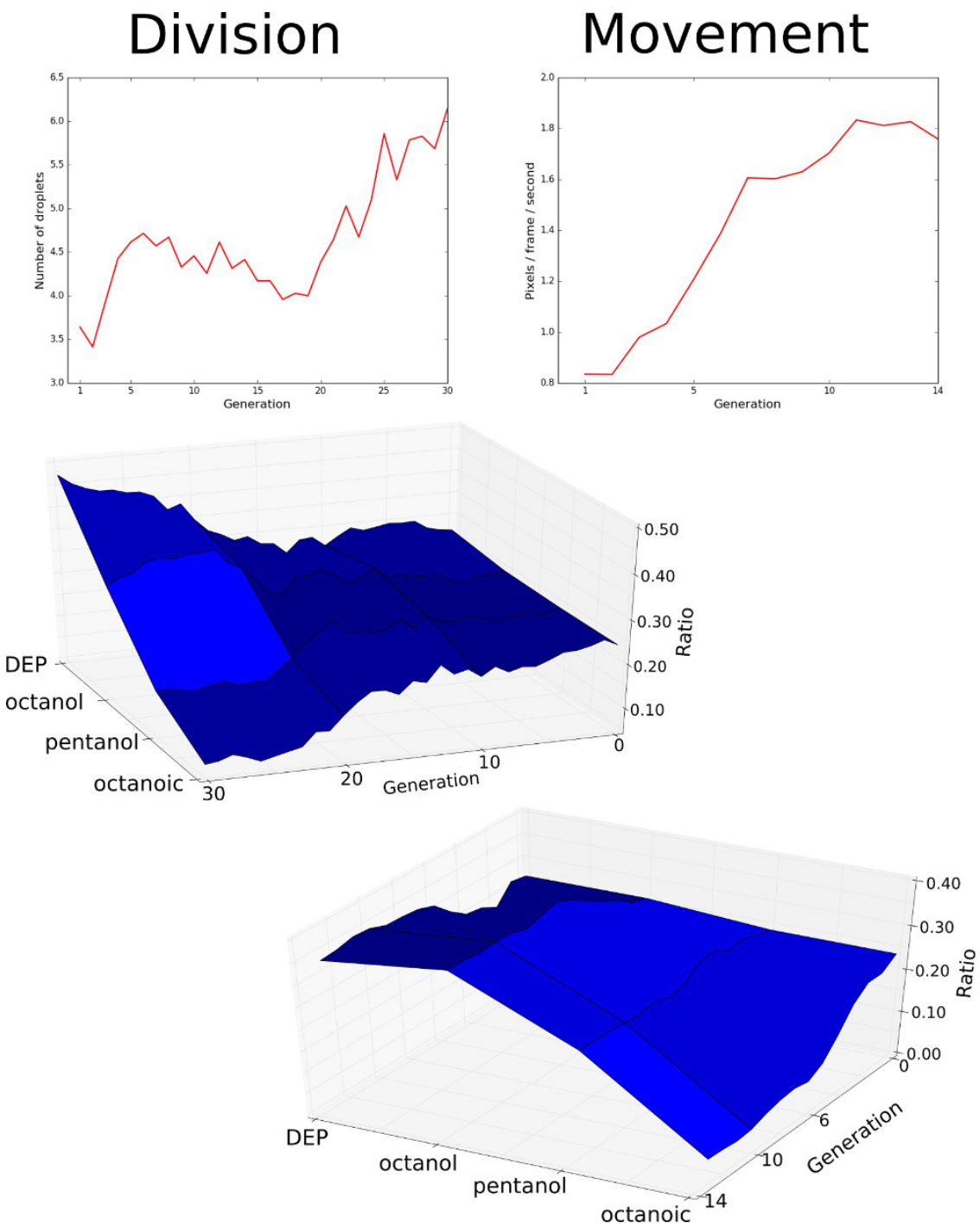


Figure 4.9: GA runs using octanoic acid. The fitness functions were division (number of droplets at the end of the experiment) and movement, which in this case was already the updated function that measures movement per droplet per frame, as define before, and that we used for all the following experiments. The "division" linear plot relates to the middle left surface, while the "movement" linear plot relates to the bottom right surface. The "ratio" label in the surface plots' Y axis represents the average ratio within a generation.

between the different diluted experiments, apart from the data being slightly skewed towards the component that was present twice. In total, the data represented here describes nine different GA runs, the three fitness functions repeated for each of the three possible dilutions. The GA parameters were set as described in Section 3.2.4 and the three fitness functions were defined as described in Section 3.2.3.3.

After the nine GA runs were finished, their evolutionary fitness trajectories were plotted, see Figure 4.10. In all cases there was a clear improvement when comparing first and last generation, although not in every case was it obvious that the evolutionary trajectory improved through all the generations, thus a question we asked ourselves was: Did the system need 21 generations[5]?

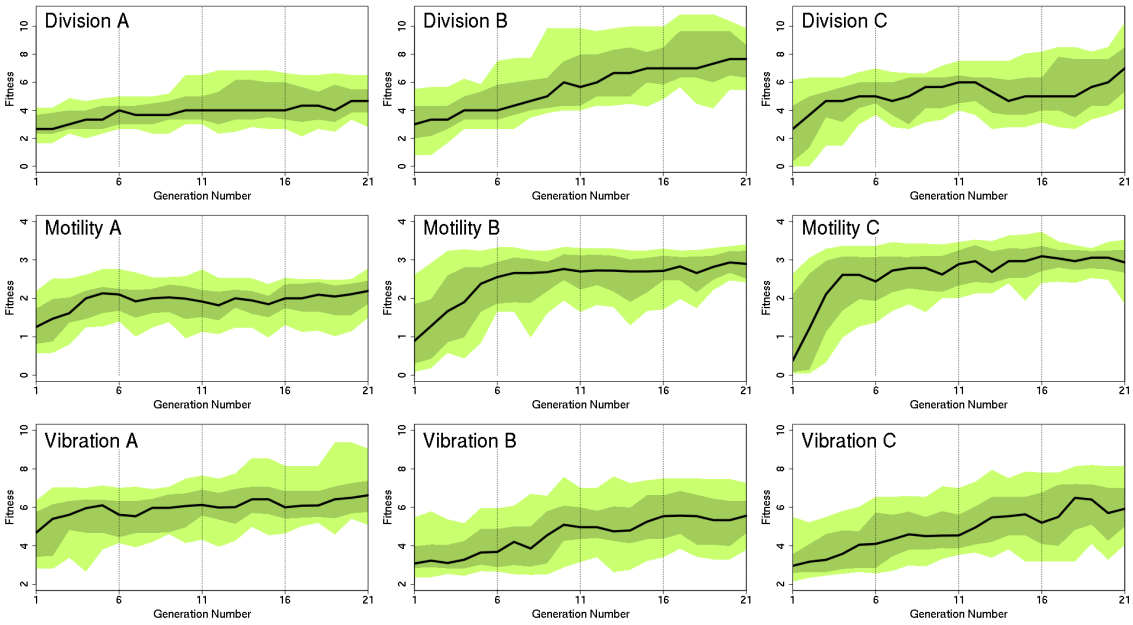


Figure 4.10: Fitness progression throughout each of the nine optimization experiments, derived from three repeats for each of the three fitness functions. Fitness is specific to each function and not comparable between functions. The black line corresponds to the median for each generation, dark green bounds the distribution between the upper and lower 25th percentile and light green bounds between the upper and lower 10th percentile.

In order to answer this question an ANOVA[6] test was performed comparing the

[5]The main reason we did 21 generations per GA run is because it took exactly one week of time, and because from the very beginning we observed that this number of generations was enough to show some sort of improvement. In AI it is usually preferred to execute an algorithm until and ending conditions is met, like for example the improvement between two consecutive generations being smaller than a set threshold. In our case we wanted all the GAs to have the same number of generations, independently of the improvement between generations, because we wanted all of them to produce a similar sized dataset we could compare later on.

[6]The "F-value" shown in the following tables is unrelated to the fitness values discussed before. "F-value" is a statistical term used when a data set is tested against a "F-distribution". The "F" comes from its full name, which is "Fisher Senecor distribution". The "p-values" as shown here were obtained using a direct transformation based on the degrees of freedom of the data set. The reason why p-values are shown here is because they are a more common statistical value.

first and last generations. For each generation only the individuals with a fitness value above the median were considered, because those ones were the most likely to propagate. The results are summarized in Table 4.1. As expected, the populations of generations 1 and 21 were completely different.

Fitness	F-value	p-value
Division	104.1	$< 10^{-15}$
Movement	74.9	1.7×10^{-13}
Vibration	43.6	2.7×10^{-13}

Table 4.1: ANOVA analysis of the first generation individuals in the upper half of the fitness distribution against the last generation individuals in the upper half of the fitness distribution. All fitness functions showed highly significant improvement in fitness from the beginning, to the conclusion of the experiment.

On the other hand, from the results shown in Figure 4.10, it was not visually obvious if population fitness showed improvement during the second half of each run. To determine if this was the case or not, the same analysis as above was repeated, but with generation 21 compared against generation 11. As can be seen in the results presented in table 4.2, both division and vibration showed significant improvement from generation 11 to the end. Movement, however, can be considered as having been optimized after just 11 generations.

Fitness	F-value	p-value
Division	5.51	0.0207
Movement	0.611	0.436
Vibration	7.08	0.00902

Table 4.2: ANOVA analysis of the 11^{th} generation against the last generation, under the same method of analysis as in table 4.1. Division and vibration both show significant improvement in fitness during the latter half of the experiment.

The same test was finally repeated for all generations, comparing its continuous fitness against categorical generation numbers. The results are summarized in table 4.3; all the fitness functions showed a highly significant result because their p-values were almost 0, meaning that the null hypothesis, which in this case was "there is no continuous growth of the fitness values through generations", can be rejected. In statistics it is usually considered that a p-value under 0.05 rejects the null hypothesis, the lower the better. In our case, as shown in the table, the p-values were lower than 10^{-15}.

Fitness	F-value	p-value
Division	407.8	$< 10^{-15}$
Movement	259.1	$< 10^{-15}$
Vibration	297	$< 10^{-15}$

Table 4.3: ANOVA analysis of all generations; the entire distribution for each generation was tested as a function of generation, expressed as a categorical variable. All fitness landscapes showed highly significant differences in variation between generations.

4.4.3.4 Fitness landscapes

With all the data generated through the previous nine GA runs, fitness landscapes were produced for each of the three different fitness functions[7] . The objective of the fitness landscapes was to create a data model of the data generated through experimentation, and estimate the gaps that were never tested by the actual evolutionary algorithms. The fitness landscapes can be seen in Figure 4.11. Because the data had four dimensions, one dimension for each chemical component in a recipe, we plotted four fitness landscapes with three components each. Although each plot is drawn in 2D, it represents three components, being the X axis one of the components, the Y axis another one, and the third component being $Z = 1 - X - Y$. While the colour is related to the fitness values, the black portions represent combinations of recipes that were not valid, because the three components must sum exactly "1". This way, for example, focusing on the first plot (top left one), a point just in the centre represents 0.1 of octanate, 0.5 of octanol, and 0.4 (1-0.1-0.5) of pentanol.

These fitness landscapes showed that all the components were present in at least one global maxima for each of the fitness functions, which was one of our main objectives. In particular, octanoic acid, which is the one we replaced before, showed a strong correlation with the vibration fitness function, as it can be seen in the sub-figure C_{iii}, although on the other hand it can also be seen that octanoic acid had a negative effect towards the other fitness functions. Another interesting result was that both division and vibration had local maxima peaks very near the global maxima. This proved that the data was nonlinear, which was a very interesting result considering that we only used four inputs.

4.4.3.5 Droplet paths

So far all the data we have produced is based on the fitness functions as described. We tried to implement other fitness functions, but we were not successful. In order to move forward we decided to try to represent the data in a completely different way.

[7]The fitness landscapes were generated by Dr. Trevor Hinkley. He used a Kernel Ridge Regression method. The full description of how these plots were obtained can be found in the Supplementary Methods section that accompanied the published publication ([Parrilla-Gutierrez et al., 2014b] - Supplementary Method 3: Construction of fitness landscapes)

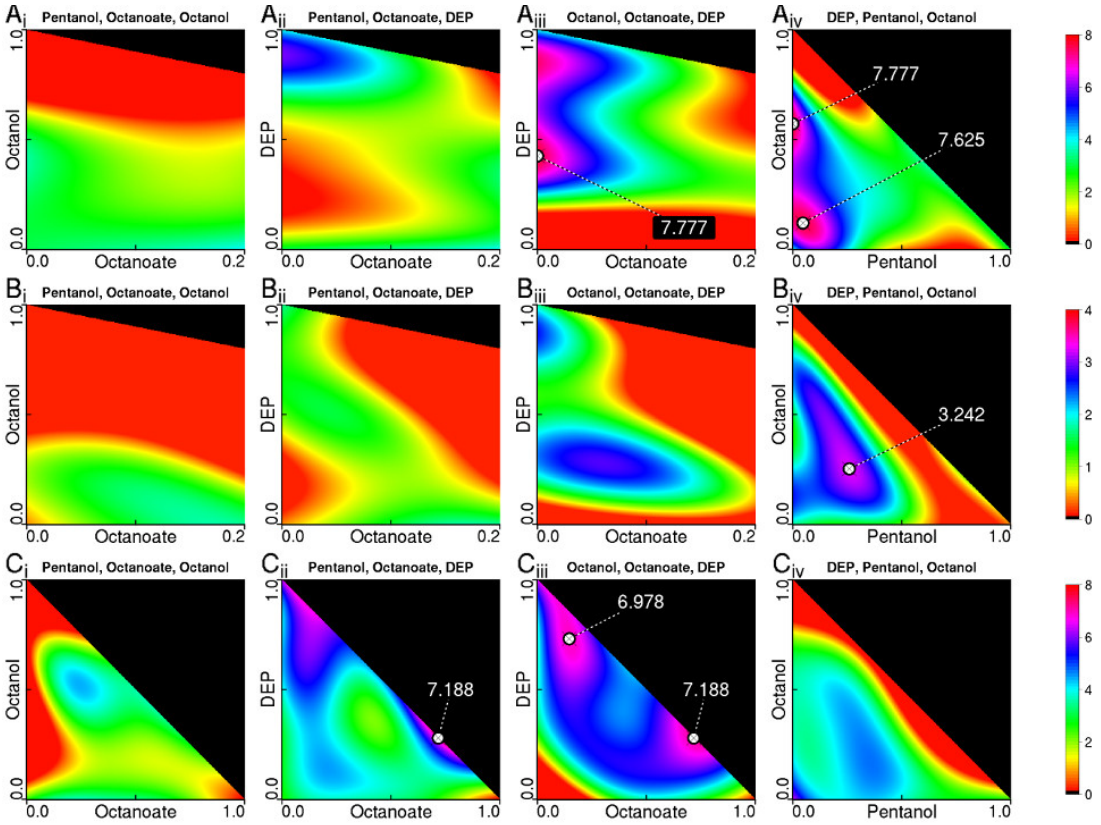


Figure 4.11: Fitness landscapes for the division (**Ai-iv**), movement (**Bi-iv**) and vibration (**Ci-iv**) fitness functions. The fitness values for each of the fitness functions are the ones described during Section 3.2.3.3: number of droplets at the end of the experiment for division (**Ai-iv**), pixels moved per droplet per frame for movement (**Bi-iv**), and angle rotated in degrees per droplet per frame for vibration (**Ci-iv**). In each plot, three substances are displayed as the plot title. The fourth substance can therefore be assumed to be held at a constant zero. Each axis shows the proportion of a substance (indicated). The proportion of the third substance (**Z**) is calculated as $Z = 1 - X - Y$. The numerical indications show the location of fitness peaks. In division and vibration two major fitness peaks were discovered. Minor fitness peaks are not indicated. The scale bar corresponds to the fitness functions for each environment and are not comparable between environments.

Other research groups have tried to extract data from similar experiments, although none of them used a robot, which severely limited the quantity of data they had to work with. We were interested in trying to replicate the work done by [Horibe et al., 2011] where the data extracted considered the motion as a whole performed by the droplets, and they plotted the paths described by them.

The three fitness functions used in the previous calculations were all based on obtaining the coordinates of the droplets for each frame, and then calculating how their positions changed over time, on a frame to frame basis. Thus, extracting similar data to the one shows by Horibe et al. was easy for us, because our computer vision library was already returning the position of every droplet in every frame. Following their results, we decided to plot the whole motion described by a droplet during an experiment, the results can be seen on Figure 4.12.

Watching the paths described by the droplets, it can be seen that they produced different drawing styles, especially when comparing the last generation. Movement created dense drawings with a lot of straight lines followed by curves. Division created small, unconnected segments. Maybe each of these segments marked the moment a droplet divided and the tracking failed for a few frames. Vibration created plots where the droplets would remain stationary and vibrate around a fixed point. The drawings created this way were only used for visual interpretation, and no numerical data was extracted.

Apart from plotting the paths, further analysis of the videos was never done, and that is one of the things missing in this project: could we categorize the behaviours using different metrics?

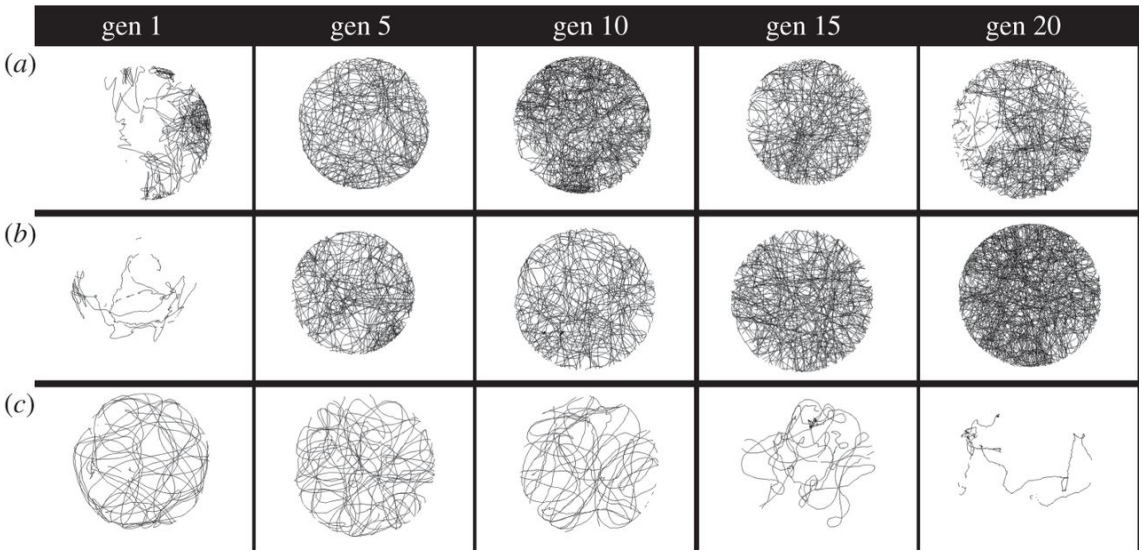


Figure 4.12: Trajectories of droplet movements in a Petri dish every five generations from the first generation (left) to the twentieth generation (right). A pixel is set to black colour if a droplet had its centre there at some point during the experiment. The value of the pixels is not cumulative: a pass is enough to mark it as black. (a) Dividing droplets, (b) moving droplets and (c) vibrating droplets.

4.4.3.6 Phylogenetic trees

Trying to find new ways of representing our data, we focused now on the results returned by the genetic algorithm runs.

In evolutionary experiments it is very common to show the progression of individuals through generations using a phylogenetic tree. An example of this technique used can be seen on [Lipson and Pollack, 2000] where the evolutionary dynamics were represented using one of these structures.

In our case, we had the perfect dataset because for each individual we were also saving its parents in the database, and each individual had exactly two of them. Therefore, we tried to represent our evolutionary data using a phylogenetic tree, but the results were not as clear as expected, and the data did not show any hidden information that we did not already have.

Figure 4.13 shows one of the phylogenetic trees generated. In this case, this tree represented one of the GA runs for the movement fitness function. The size of the nodes is related to their fitness value (a higher value would generate a bigger node), and it can be seen as expected that small nodes did not propagate into successful offspring. It is also interesting to see here how a few of the parents seemed to dominate most of the mating pool and generate a big quantity of offspring.

Another phylogenetic tree was generated using one GA run for each of the three different functions. All the individuals were added to a set, where only information about their parents was specified, thus there was no information about individuals belonging to the different fitness functions. All the nodes had the same size, and as before each node had an arrow to each of its parents. The difference now is that the colour of each node represented its composition, relating RGB to pentanol, octanol and DEP. This way, if an individual only contained pentanol, its colour would be red, and if it was half octanol and half DEP, its colour would be cyan[8] The phylogenetic tree outputted three clearly separated branches, thus it was able to cluster the individuals based on its fitness function. Although this result was interesting, any simpler clustering algorithm would do the same.

A potentially interesting study based on this phylogenetic tree would be to analyse the individuals within the different branches, and check if all of them belonged to the same fitness function. Thus, to check if the tree was actually clustering the nodes in the right way. If the clustering was not perfect, it would be interesting to study the individuals that clustered within another fitness functions, because overlapping of fitness landscape was already shown in previous results, and it is an interesting feature common in more complex systems. This could potentially answer one of our ultimate goals, which is, could we evolve a droplet to A, then B and then C? Could we evolve droplets to maximize different fitness functions? A droplet that was optimized based on a fitness function A, and then was clustered with droplets optimized for another fitness function, would be a good start to answer this question.

[8]Green and blue combined using the same ratio generate cyan.

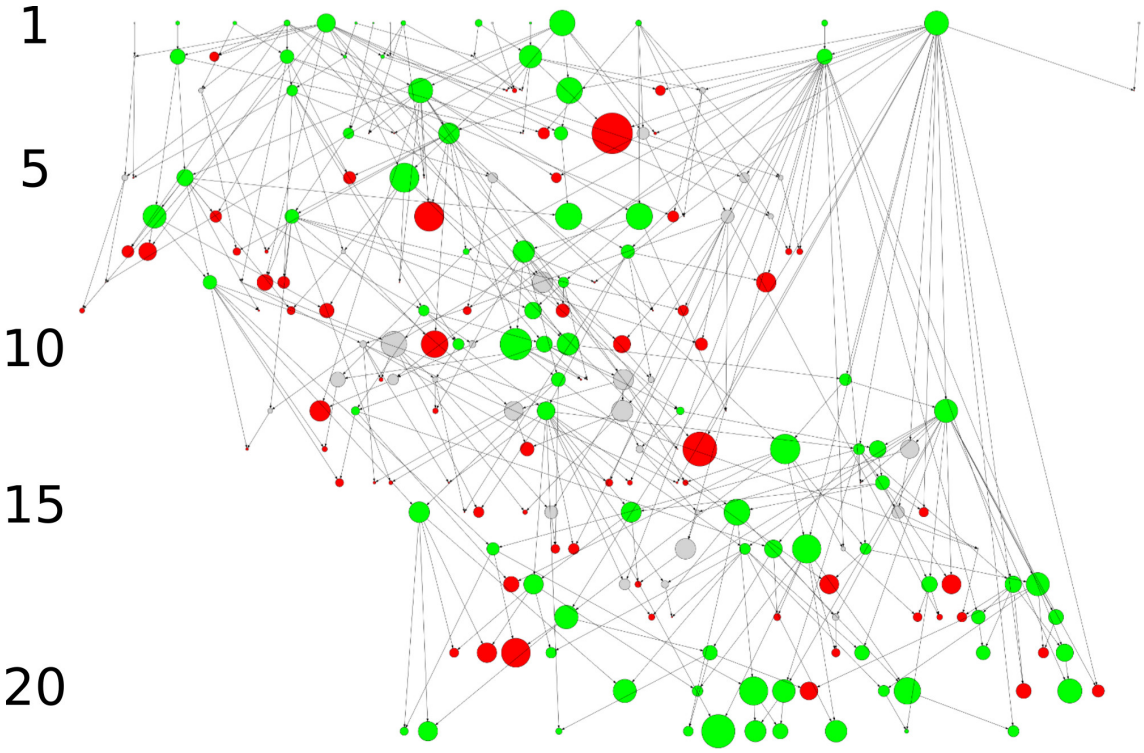


Figure 4.13: Phylogenetic tree for the movement fitness function. Each row represents a generation as marked by the left column. Each node is an individual, and each arrow connects an individual to its parents. Green colour means that the node was a leaf in the last generation, or that one of his offspring would eventually become a leaf in the last generation. Red colour means the node was never chosen as a parent. Grey colour means the node had offspring, but none of them were part of the last generation. The size of the node represents its fitness value, the higher the bigger.

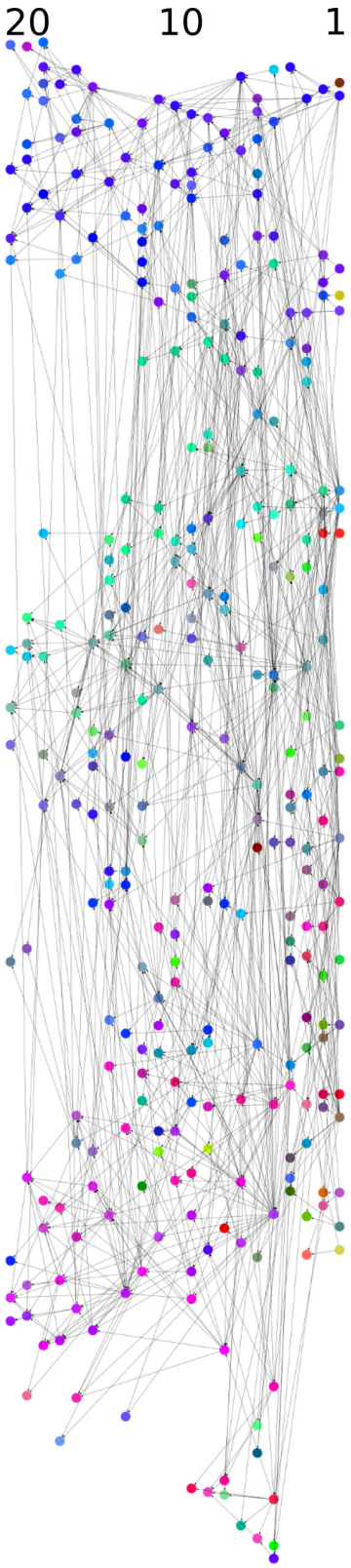


Figure 4.14: This phylogenetic tree is placed sideways in order to fit it here. The right column would be the first row or genereration. It combines the three fitness functions. Each node represents an individual. The colour of each node is directly related to the individual's composition, being RGB as pentanol, octanol and DEP.

4.5 Discussion

Through this project we were able to build from scratch, quickly and cheaply, a robotic platform whose design was based on an experimental specification. In order to do so we used 3D printers as rapid prototyping tools, and all the other components used can be easily found in any hardware store. The use of these new tools allowed for extreme customization of our robot, targeting exactly the experiment we wanted to develop. At the same time, it would be very easy to change some of its parts in order for the robot to be suitable for other experiments. We won't claim that our robot was the first ever built following this approach, but we were probably among the first ones.

We were also able to pair our robot with a genetic algorithm in order to optimise the described experimental specification. The use of AI in chemistry is not new, and there is plenty of literature about using algorithms to optimise reactions. We consider that the interesting twist in our project was how chemistry and robot were coupled, in a process similar to robotic embodiment. The main criticism this work received was negating this "twist", and saying that the robot is just another automated platform, which was not adding anything of value to the experiments apart from pure automation, and that the coupling with chemistry has already been done before. It would not be fair to argue with the critics without giving them a chance to respond back. The only thing we can say is that their criticism served as inspiration, and that the project named "Flowbot", which will be described in the next chapters, is a direct answer that tries to solve some of the problems they enunciated.

From an origin of life perspective, we showed that simple oil-in-water droplets were able to undergo an evolutionary process[9]. This was interesting because our droplets did not contain any sequential information, like DNA or RNA, and all the information was contained in its composition. Critics, again, would say that evolutionary algorithms have been used before with chemistry problems, and that simply using a genetic algorithm does not directly relate it to any origin of life theories. That statement is not completely true, because a lot of different systems like economy and/or Twitter ([Oka et al., 2014]) are already studied from an artificial life perspective. In any case, our system was chosen to mimic what is already described in the literature as a protocell, and in particular our formulation focuses on surfactant molecules, which are the ones that are responsible for creating the required compartmentalization in any known cell. Moreover, our research focused on optimising a series of traits that are known for any living entity: division and motility. Therefore, while just the coupling with a genetic algorithm does not make a system viable from an origin of life perspective, we think that our system was already very similar to possible described candidates, and we showed that it is indeed able to undergo an evolutionary process.

The ultimate objective of this whole project is to create an autonomous evolvable chemical system. By this we mean to obtain the results as described in this chapter but without a robot[10]. The obvious problem is that without a robot, in our particular

[9]Some may consider this an optimization process instead.

[10]By robot here we mean the union between software and hardware

case, there is nothing, just an empty Petri dish. We think, though, that this project has set the foundation to pursue this objective, because at least it got all the different elements working, and it will allow us to follow a top-down approach, by removing, in successive iterations, elements from the robot, embodying them instead into the chemical system. A first step in this direction is the project "Flowbot", which is described in the rest of this thesis.

4.6 Future work

Even though the Dropbot project was finished once the article got published, the team continued to develop the concept of merging robots and chemistry, making this section about "future work" easy to write, because it can be said that at the moment of writing this document it is "present work". Figure 4.15 shows the different projects that continued the work described here.

"A", also known as "Surfbot" is the direct successor. The robot frame was redesigned to be more compact and robust, and the automated syringes were also redesigned to minimise its free movement and increase its accuracy. The advantages of this project are (a) on one side that the number of inputs was increased from four to eight, increasing the number of aqueous phases, and (b) on the other side, that it was prepared to hold more than one Petri dish, allowing extra time between re-usage of Petri dishes to let the acetone evaporate, minimising possible contamination. This project was developed by other members of the Cronin group, and it is unrelated to this thesis.

"B", also known as "DropletFactory", is the last iteration of the project, and its main aim and advantage over the other robots is that it is designed to execute experiments in parallel. Both in Dropbot and Surfbot the main three phases of an experiment are: prepare reactants, execute experiment, and clean dish. These phases were executed in sequence, thus while the robot was busy doing one of them, it could not do anything else. DropletFactory divides this process in different hardware stages, therefore it can prepare, experiment and clean at the same time, reducing the time required for each experiment from around 5 minutes to 1 minute. This project is being developed by other members of the Cronin group, and it is unrelated to this thesis.

"C", also known as "EvoBot", is a robot created by researchers in the university ITU in Denmark as part of a european project the Cronin group is part of, called Evobliss. Its main difference with the other ones is that it is not doing droplet chemistry, but it is used to develop microbial fuel cells.

"D", also known as "Flowbot", is a 3D printed piece that aimed to encapsulate all the functionality described in this chapter in a single monolithic device. This project was developed by us, and it is going to be deeply explained in the following chapters of this thesis.

There is still some functionality that was on the first theoretical design we did four years ago but that was never implemented. The main functionality we still discuss about within our team is the "droplet catcher". The idea is to be able to dynamically remove droplets from a running experiment, or to dynamically add new components to an existing droplet. This is very similar to what is already described

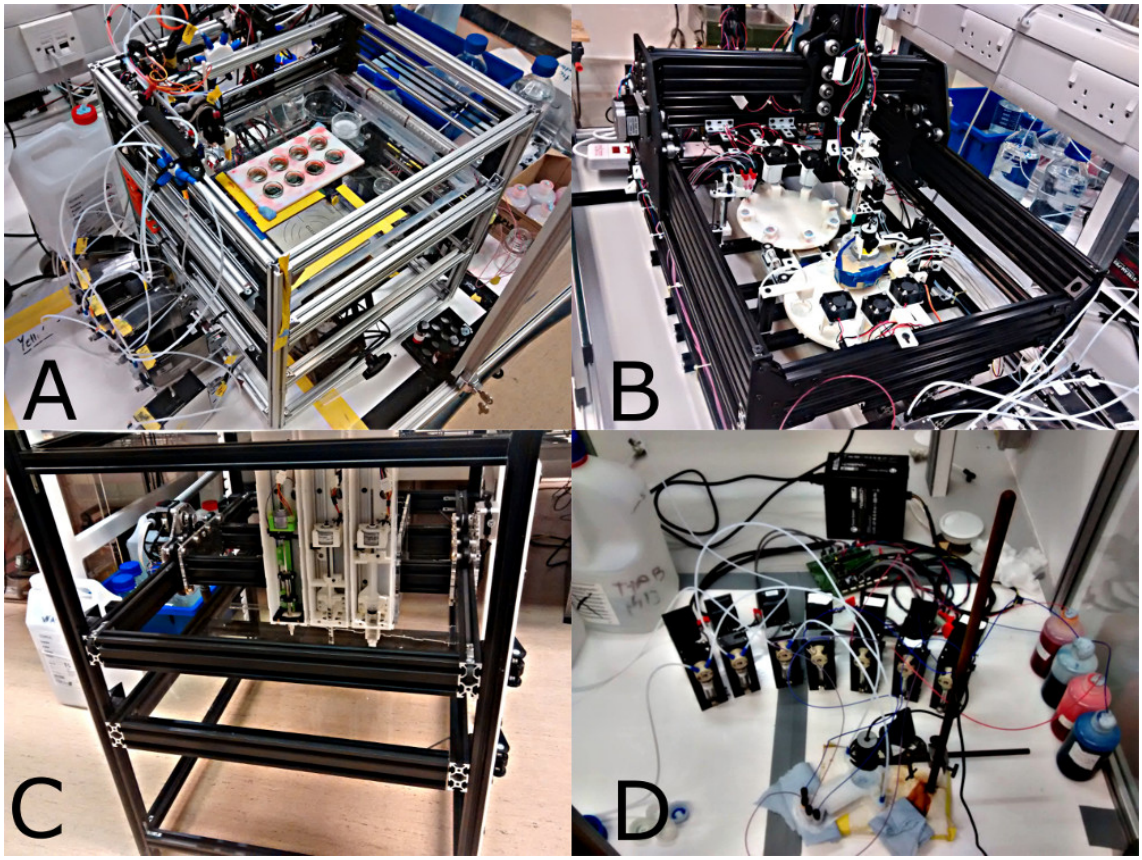


Figure 4.15: Robot projects that started as successor of this project. **A:** Surfbot, **B:** DropletFactory, **C:** Evobot and **D:** Flowbot.

in [Hanczyc et al., 2014], but so far it has never been implemented in any of the robots described here. The other functionality we still discuss about within our team is adding a microscope, like the one described in Section 3.2.5. It would be also very interesting to add other types of sensors, like a pH meter or any equipment that would be able to provide more data about the experiments.

Another objective slightly different would be to continue exploring the chemical system, and in particular increase its complexity. There are many things that could be done, but we will focus on creating libraries of droplets that would be added into a Petri dish and create interactions between them, see Figure 4.16. This might happen without the help of the robot, but the robot could also alter the aqueous phase, or add some sort of chemotaxis in order to force the droplets to collude and fuse, generating a new droplet with different characteristics. A similar idea would be to consider each droplet as a small reactor, and by forcing them to fuse, a chain of reactions could be controlled within the droplets. Another possibility would be to consider this fusion process the fitness function of another genetic algorithm, and consider a different set of inputs until it happens.

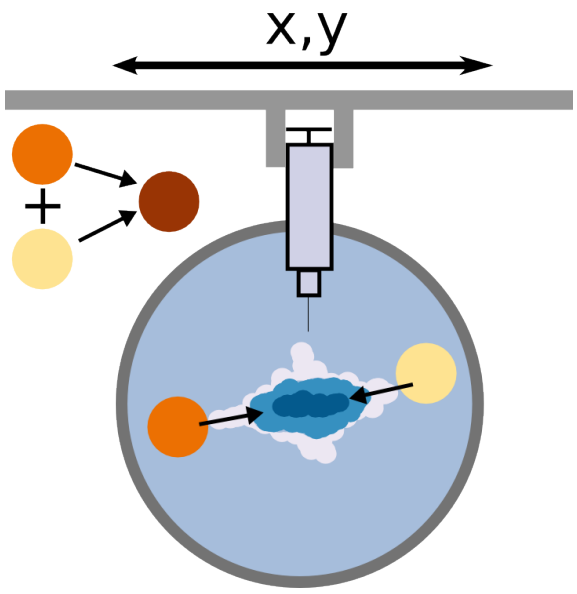


Figure 4.16: Droplets with different characteristics could be added to a Petri dish, and the robot could create a chemotaxis trail, forcing the droplet to merge and create a new one with different characteristics.

5

Flowbot platform

Evolution is an autonomous process governed by natural selection, which is guided by an evolutionary fitness that measures the quality of an individual against the rest of the population and the environment, and selects the fittest ones to propagate their traits over successive generations. Although evolution has been deeply studied in very different fields, such as biology or computer science, the impact of rapid environmental changes has never been studied in a real world artificial system. In order to explore how the environment defines and modulates the evolutionary fitness, we embodied evolutionary fabrication into an automated platform where the evolutionary dynamics of populations and the environment are explored in a single system. Herein we present a robotic platform where evolution is embodied into a self-contained 3D printed device able to generate populations of physically interacting chemical entities. The device consists of a network of flow channels where four different oils are mixed, and an in-built and designable evolutionary arena where droplets are generated through a capillary like channel. Meanwhile, a camera records the behaviour of the droplets as they come in contact with the aqueous phase, and image recognition software is used to describe their activeness, which is used as fitness criteria to direct an evolutionary algorithm. Using designable environments we were able to physically modify the selection pressure during the execution of evolutionary experiments, resulting in very rich fitness landscapes, akin to natural evolution. Our platform is both flexible and can act as a sandbox where to test from origin of life theories to chemistry formulation experiments.

This and the next chapter comprise all the work done for the project known as "Flowbot". The first chapter focuses on the platform itself: how it was designed and built, while the next one will focus on the results obtained. The project received this name because its core idea was to develop a platform very similar to the robot just described, but using flow processes instead of batch ones. The work itself was completely unrelated to the field known as "flow chemistry". It was probably more similar to some of the work done with millifluidic devices (see thesis' introduction p.49, Section 1.5.2).

5.1 Introduction

The work presented in this chapter was heavily influenced by the previous project ("Dropbot") and by some of the other work that was being developed at the same time in the Cronin group. The objective of the Flowbot platform was to replicate the functionality of a liquid handling robot, but using a 3D-printed device instead.

5.1.1 While Dropbot was being developed

Before this PhD started the Cronin group had already specialized in the use of 3D printing technologies and chemistry. In particular, a paper was published [Symes et al., 2012] where a reaction vessel was 3D printed to aid organic and inorganic reactions. The most interesting result was that, aided by the use of 3D printers, the architecture of the vessel could be modified, making different reaction products. This gave birth to the term "reactionware". The polymer used during that research by the 3D printer was from the silicone family. Although it was good enough to print devices, it had a very low chemical resistivity, and this limited its possible chemical applications.

The Cronin group continued to develop the technology of integrating chemistry and 3D printing (see thesis' introduction p.52, Section 1.5.3). In particular, relevant to the development of "Flowbot" was the use of 3D-printing to manufacture millifluidic devices using polypropylene (PP) [Kitson et al., 2012], see Figure 5.1. PP is a very interesting polymer because it suffers no degradation to any of the compounds used during the Dropbot project, like acetone, alkaline solutions and fatty acids. The devices they manufactured there were also very interesting because they contained channels very similar to the ones used in microfludic devices, but on a bigger scale[1].

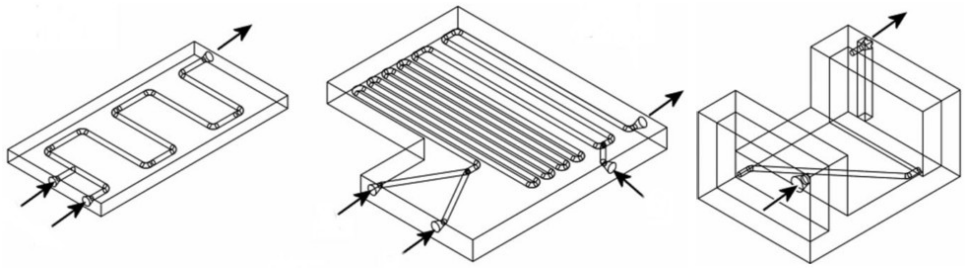


Figure 5.1: Reactionware project that inspired Flowbot. The devices here shown are very similar to the ones used in microfluidic devices. These ones were printed using a 3D printer and with PP as polymer. From [Kitson et al., 2012]. The arrows mark the inputs and outputs. Courtesy of the Cronin group.

When this PhD started there were two different projects running in parallel, both of them were based on a different platform aimed to perform similar experiments. One of them was Dropbot, as explained in the previous chapters, the objective of which was to build a custom liquid handling robot to generate populations of

[1]They described channels with a diameter of around 0.8 mm. Microfluidic traditionally channels have a diameter of 0.1 mm or less.

droplets and evolve them using a genetic algorithm. The other project was based on using a microfluidic device to also generate droplets and similarly evolve or optimise them using some sort of algorithm.

The microfluidic based project never produced the results expected, mostly because the manufacturing processes of this technology are very complicated, and the number of operations required were very complicated to implement in a single device. Figure 5.2 shows the theoretical design. The main difference with what was explained during the Dropbot chapters, is that while before we only saved the recipe formulations, this microfluidic device aimed to save the droplets themselves, and re-circulate them through the device.

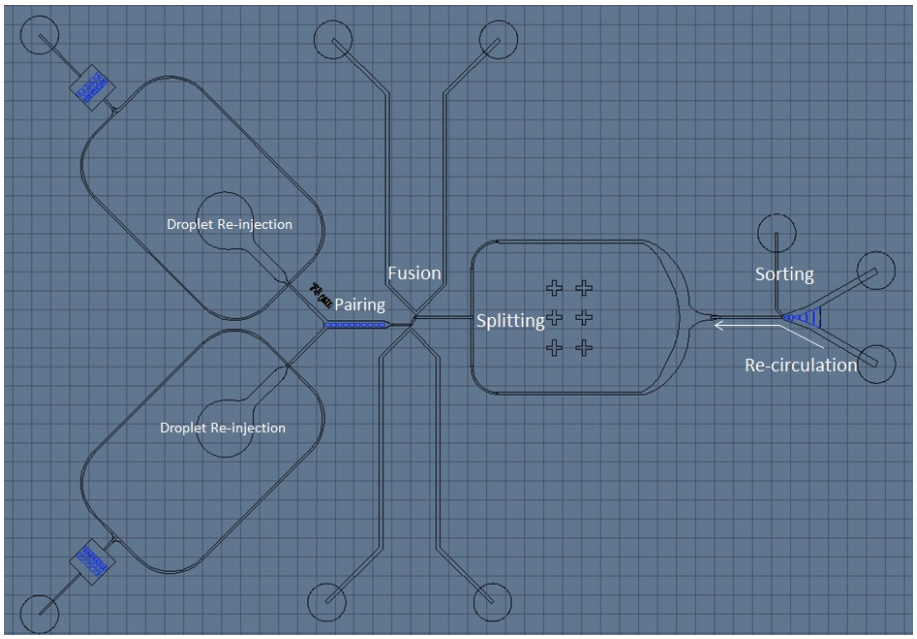


Figure 5.2: Surfactant evolution microfluidic device. This design was never fully implemented. While Dropbot worked with oil formulations, this device received pre-generated droplets, which were fused and then tested. Another difference is that the tested droplets were supposed to be kept, and then re-circulated as inputs and used again. Image courtesy of the Cronin group.

The inspiration for the project that will be described in this chapter came from the idea of mixing the theoretical designs from the microfluic based project, with the "reactionware" manufacture using a 3D printer and PP as polymer. Obviously, with a standard 3D printer it is impossible to print true microfluidic devices, and we never tried to do so. Instead we focused on replicating some of the functionality in a *millifluidic* scale[2].

5.1.2 Development from Dropbot

Some of the general criticism we received about the Dropbot project was:

[2]By millifluidic scale we mean channels with a diameter of around 1 mm. This scale is perfectly feasible with standard 3D printers.

- It is just automating a chemical process, and that has already been done several times.
- The robot is doing everything, nothing is "embodied".
- Those are just oil droplets, not protocells.

About the first point, a big part of the chemistry community only care about pure chemical advances, so no matter what we did, we would never appeal to them.

About the second point, what they meant is that the robot was meaningless, because the same operations could have been performed by a researcher, or by any other similar robot[3]. We partially agree with them, and comments like that were one of the main motivators for this new project.

Finally, about the third point, what they were meaning when they said that our system was not a protocellular one, is that our project could not be linked to the origin of life, and that we were just optimising some sort of chemistry. We will not be able to address this point in this chapter because the new project will focus entirely on building a new platform while the chemistry will stay the same as before (see Section 4.2, p.86).

5.1.3 Conceptual limitations of liquid handling robots like Dropbot

Once the Dropbot project was finished we had two possibilities: continue iterating it, or starting something different but in the same line of research.

Continuing to iterate the Dropbot project meant not doing anything conceptually new, and the three main criticism points described above would prevail. We could design a more stable robot, faster, cleaner, with new functionalities, new sensors, new inputs and outputs, parallelize its operations, obtain more data, increase the throughput ... The reality is that none of these features meant any significance advance to the key questions we wanted to answer[4]. They were just improving the platform. It was not science, just engineering.

Therefore, we decided to focus instead on the Flowbot project, which offered a slightly different platform that tried to answer some of the criticism we received before. We shared some of that criticism, so we were not just answering to people from the outside: we were also answering to ourselves.

5.1.4 From Dropbot's paper discussion section

It might seem that the idea of the current project appeared at some point during the end of the Dropbot project, once we obtained some results and received feedback. The reality is that right from the start we had in mind something similar, from a conceptual point of view, although we did not know at that point how to implement it. In particular, from the Dropbot's paper discussion section:

[3]There were no similar robots when the Dropbot project was finished.

[4]Already explained during the thesis' aims and the Dropbot chapters. The main question of this thesis is "can we create an autonomous evolvable system?" which we reformulated as "Could it be achieved following a top-down approach?"

> "While most of the characteristics of 'life' are wholly dependent on the robotic adjunct, in the current system, we propose that these dependencies could be removed stepwise, in serial evolutionary experiments, to move the chemical system towards full autonomy, avoiding the all-or-nothing barrier that currently plagues the study of minimal chemical replicators. It is therefore our intention to focus follow-up work towards the creation of artificial life-like assemblies, driven by a thesis that is chemically agnostic, thereby opening up investigations of new evolvable chemical systems."

Our ultimate goal is to create an autonomous and evolvable chemical system. A system that could replicate the results from the previous chapter, but without a robot, because there were no robots four billions of years ago assisting life and evolution.[5] We are very far from this point, but we consider Dropbot a first step: a simple platform that could emulate evolutionary processes using viable candidates for the origin of life. There were two possibilities about how to move forward: (a) we could keep the same platform, but add more to the chemistry, or (b) we could keep the same chemistry, but remove bits from the platform. Point (a) would be the classic bottom-up approach to the origin of life, as explained during this thesis introduction, while point (b) would be a top-down approach to the platform. Because the core of the people working on this project were not chemist, but engineers, we decided to move forward using the second approach: How much of the platform can be remove while keeping the chemical system intact and working? Could we remove all of it?

We believe the Flowbot project is one tiny step in this direction, while continuing in a platform like Dropbot would not be a step forward in either direction.

It is worth mentioning that our focus was not only about emulating the origin of life, but that we were also interested in creating artificial life forms, from a standard robotic perspective. We considered the oil droplets basic artificial life entities, and our ultimate objective was also to create autonomous and evolvable artificial life.

This problem was approached by focusing on how life emerged on planet Earth, and considering if artificial life could emerged replicating such processes. In particular, our focus during this chapter is on robotic embodiment, therefore the question while designing the new platform was: "could abiogenesis processes be embodied into a robotic platform?" Most of the work developed nowadays in the artificial life field goes in the other direction, and researchers instead of searching answers in the past, try to find them in the future, and that is why the classic humanoid robots have nothing in common with any abiogenesis process.

5.1.5 What could be removed from Dropbot?

Our custom liquid handling robot was designed to mix liquid volumes into well plates, and then generate droplets of the mixtures in a Petri dish, which had been previously loaded with aqueous phase. All these operations were performed using automated syringes attached to a carriage that could move around the X and Y axes. The core idea behind Flowbot was to encapsulate most of Dropbot's operations in

[5] As far as we know.

a single monolithic device. Our design was based on how similar operations are done using microfluidic devices, where mixing is done using serpentine channels, and droplets can be generated using capillary or T-junctions, among many other techniques. By encapsulating all these functionalities into our device, we technically removed them from Dropbot, so we took a step into removing elements from the robot. Some may argue that the elements are still there, but at least we can argue that the active elements were transformed into passive ones.

Not everything could be removed, because for example we still needed syringe pumps in order to transport liquid into the device. We also needed a set of electronics to control the pumps, and a computer which was connected to the platform and controlled the experiments. The chemistry was still very far away from being disconnected, although we think, as said, that we took a small step in that direction.

5.1.6 Embodying evolution

During the previous chapter we showed that populations of oil in water droplets underwent an evolutionary experiment by coupling their compositional information with the "proto metabolism" described by the semi permeable interface. We also showed that this coupling between the droplet compositional information, which we called recipe or formulation, and the actions happening across the interface is what caused the droplets to describe a series of behaviours that were used to direct a fitness function through an evolutionary experiment.

However, evolution is not just a gene-driven process, but it is formed by the interplay between the genome, the cellular environment and the external environment. During this project we tried to take a step forward into protocelullar evolutionary experiments by creating a platform which enabled us to embody programmable environments in order to fulfil the described interplay required by evolution. Using the programmable properties of our device we are able to perform modifications in the environment configuration of our experimental arena, allowing us to study the adaptive evolutionary dynamics of simple oil-in-water droplets. See Figure 5.3.

Using CAD software and 3D printing techniques we aimed to design and manufacture reactionware that could embody an evolutionary experiment based on a genetic algorithm which at the same time is a discrete model of natural selection. The use of 3D printers allowed us to include the device itself as part of the evolution, which permitted us to study the coevolution between the entities and the environment they reside in. Taking advantage of these new prototyping techniques we were able to use evolutionary operations embodied in our design, permitting us to alter its structure while the experiments were running, emulating how populations of entities respond to sudden environmental changes, and how these changes act as a filter which reshapes natural selection based on the interaction between the environment and the behaviours described by the population of droplets.

Nevertheless, it is important to note that this evolutionary embodied design is not only restricted to the study of evolutionary dynamics, but it can be easily extended into the study of other complex chemical systems, offering novel means into the exploration of complex reaction spaces, enabling to control the interplay between robot automation, artificial intelligence and chemistry.

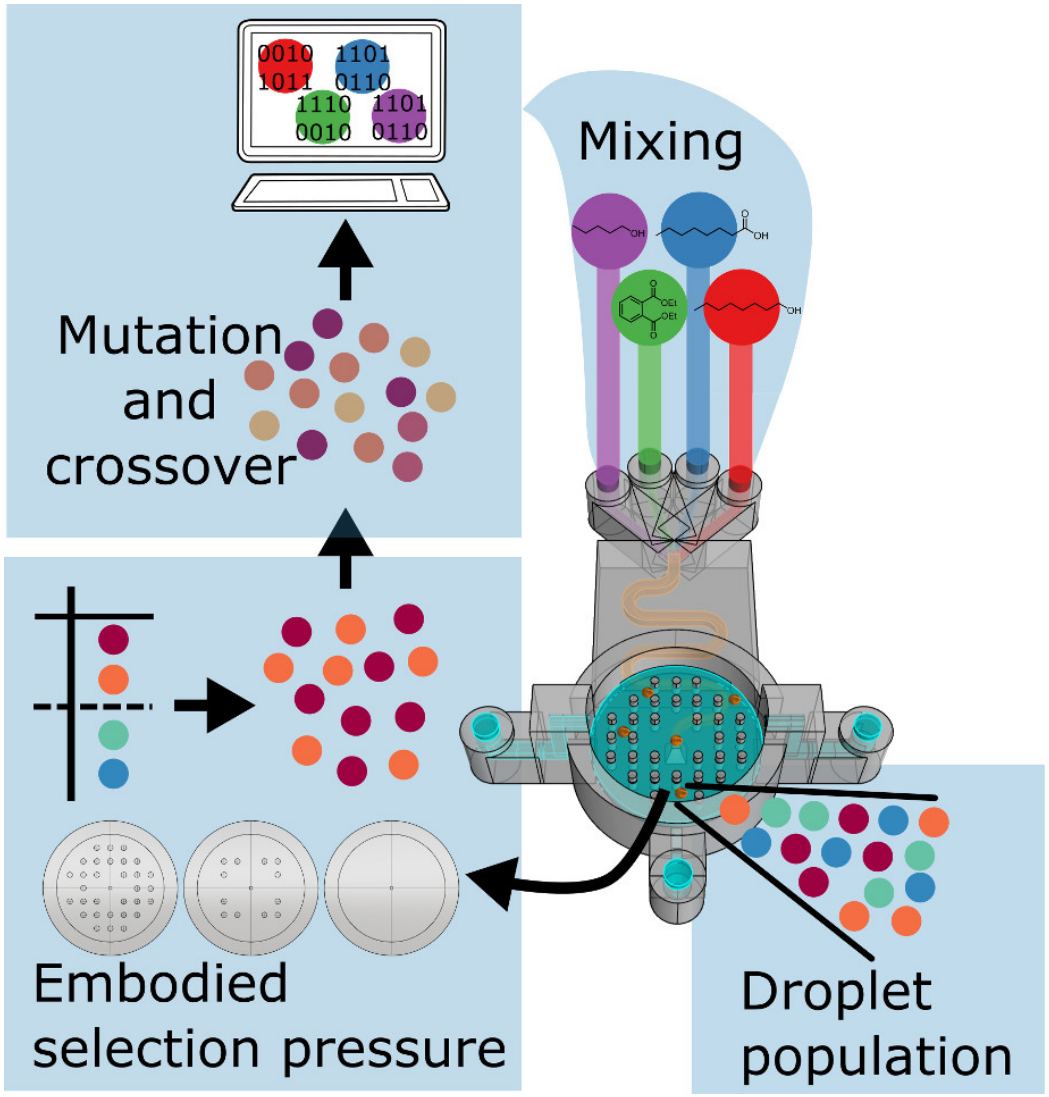


Figure 5.3: Schematic describing the embodied evolutionary process. In the first step, a computer generated random formulation recipes using ratios of 1-octanol, octanoic acid, 1-pentanol and diethyl phthalate. These oils were mixed through a serpentine channel, and populations of droplets were generated into the evolutionary arena. The droplet behaviours were then analysed, and ranked using a defined fitness function which relied on the programmable arena. The best ones were selected, and new droplet formulations were generated after "random" and "crossover" operations. This process continued through iterated generations.

We think that by embodying both functional elements like a mixer or droplet generation, and conceptual elements, like parts of a genetic algorithm, we addressed some of the criticism the previous project received. We also think that by performing this embodiment process we were pushing forward the idea described in Dropbot's paper discussion section, where we suggested that dependencies could be removed stepwise, in serial evolutionary experiments.

5.1.7 Chapter objectives

As said, the Flowbot project is split into two different chapters. This one will only focus on the platform itself, thus the project objectives will be only about the manufacturing process, ignoring the results obtained.

- Design and manufacture a single monolithic device that can generate populations of droplets into an arena.
- Add a serpentine mixer, and test that four different inputs can be mixed.
- Add a cleaning cycle, and test that the device can run in a continuous unattended way.
- Code the software to control all these processes.
- Add a camera to record the experiments, and code a image processing algorithm to analyse it.
- Test the robustness and possible experimental degradation of the device.
- Discover new ways of creating new environments.
- Manufacture these new environments into the device.

5.2 Designing the device

Chapter 3 described the design and functionality of a basic liquid handling robot constructed and used to generate surfactant based experiments. As happens with many other similar robotic platforms, especially the ones doing chemistry or similar disciplines, its workflow could be divided into three steps: prepare experiment, execute experiment and clean experiment. How these steps were performed was detailed in said chapter.

To sum up:

- The robot was a cartesian machine, therefore it had a tool that moved around the X-Y plane. In our case, the tool was a carriage that contained a syringe powered by a servo motor and a series of tubes connected to liquid syringe pumps.
- In fixed positions over that X-Y plane there were also a 96-well plate and a Petri dish.

1. The experiments were prepared in the 96-well plate where each well contained a magnetic stirrer bar. The robot would move the oil outputs over a defined well plate, and dispense the oils in different ratios as defined by the recipe being tested.

2. In order to execute an experiment, the robot would first fill a Petri dish with aqueous phase, then absorb some of the previous oil mixture using the syringe, and then place four droplets in the aqueous phase. The experiment then was left for a minute, and a video was recorded.

3. In order to clean the experiment, the Petri dish was cleaned with a sequence of washes using acetone and aqueous phase. The contents of the Petri dish were removed after every step using an output tube.

The first objective of the Flowbot project was to replicate this exact process into a single monolithic 3D printed device so that it could emulate the functionality of a basic liquid handling robot. In order to do so, a prototype based incremental design was applied. Thus, the aim was to have a working prototype as soon as possible, even if that prototype could not replicate most of the functionalities of a liquid handling robot. Once a basic prototype worked, we would add new features one by a one, testing them until the prototype was fully working again, and then iterate it adding a new feature again.

In this section the focus will be about designing new functionalities and not about how some parts of the device were redesigned in order to overcome manufacturing problems. Thus, it will be assumed that each prototype was successfully manufactured. Section 8.2.3 covers the manufacturing problems and how we worked around them to get the devices working.

5.2.1 Droplet generation

The ultimate objective of this project was to perform experiments where populations of oil droplets were generated into an aqueous phase. Therefore, the action itself of generating droplets was the most important one and the first to be prototyped.

As explained in Chapter 3, in the previous project populations of droplets were generated with a syringe. The syringe had a servo motor attached to the plunger that moved it using small pulses in order to release small quantities of liquid in the shape of droplets. This is how a human would make droplets using a syringe.

In the new device we wanted to replicate a similar functionality in a flow fluidic device, and our main inspiration were microfluidic devices, because a big portion of their research is based on droplet experiments similar to ours. As explained in Section 1.5.2, the three main ways of generating droplets in microfluidic devices are: capillary, T-junction and flow focusing. Nowadays the most used methods are T-junctions or flow focusing because they adapt better to the single layer standard manufacturing process, and because in general they produce better results, especially when you need to control the size of the droplets. The main problem of the capillary option is that it is based on having a channel inside a channel. In single layer monolithic devices this can be very difficult to achieve.

Our first concept design was based on using a T-junction, but after drafting how the whole device was likely to be, especially the need for an open arena to perform experiments similar to a Petri dish, we decided to try a capillary like design first because its design and implementation were easier. The objective was to design the arena with a capillary-like outlet in the bottom. In this way, the arena would be initially filled with water, and then pulses of oil phase would be generated into it through the capillary-like outlet. It was not exactly like a capillary droplet generator in microfluidic devices, but it follows the same idea of a phase flowing inside the other phase. See Figure 5.4.

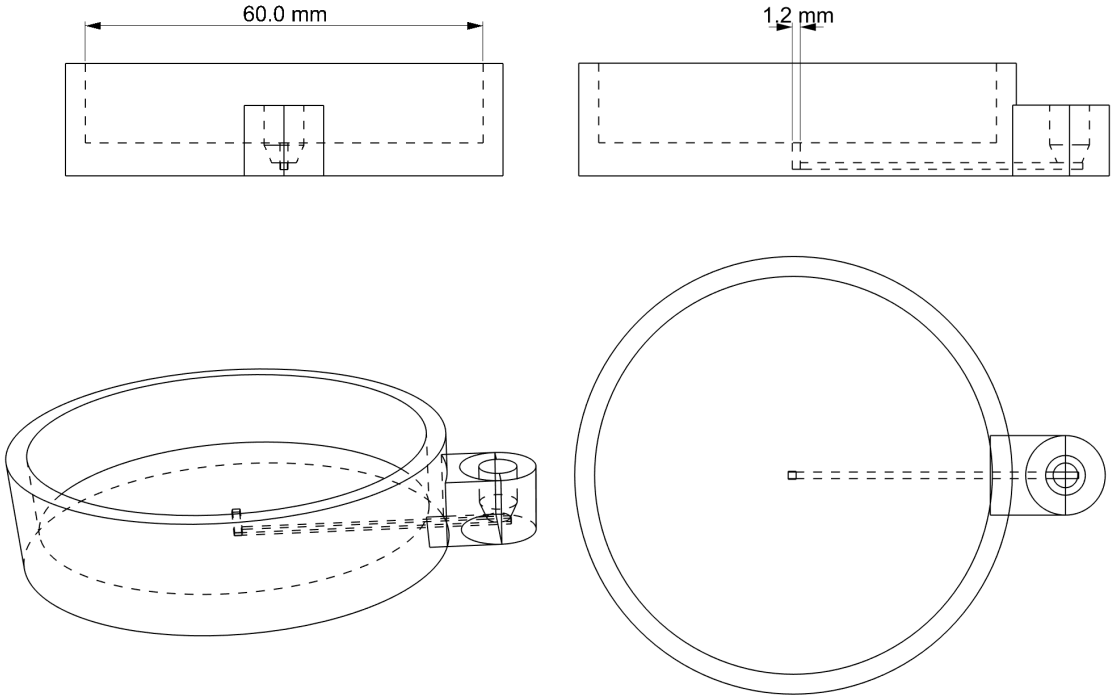


Figure 5.4: Flowbot first prototype. It only contained one channel to input the oil phase. The aqueous phase was added manually, and the cleaning was also done manually.

This first prototype only contained one input which was used for the oil phase. The aqueous phase was added manually, and the device was also cleaned manually. The prototype aimed to answer three questions: Could it generate oil droplets? Would the internal channel leak? Would the tube-fitting connecting the pump and the printed device be sealed?

Starting with the last question, the device had a 6 mm diameter hole where to insert the fitting, and this hole was 3D printed as part of the device. Thus, it was very important to test that the fitting would fit inside and be sealed, especially when pumping high pressure liquid. Our initial tests were successful.

Regarding the second question, the device inner channel did leak in this first prototype. Inner channel's leakage is a problem we have not fully solved. Leakage problems are discussed in more detail in Section 8.2.3 (p.186). From now on until the end of this section we can assume that the devices never leaked.

Finally, the device did manage to generate droplets, although there were two drawbacks. The first one was that more often than not the droplets would accumulate in a heap above the outlet point. This could be fixed by generating faster pulses, but depending on how fast they were it might generate more than one droplet. The second drawback was that the droplets were bigger than expected, between 5 and 10 µl. This was actually very similar to the droplets we generated with Dropbot, but bigger than we expected. The outlet point had a square shape with 1.2 mm side in the CAD design, although the 3D printed hole was smaller than this[6] because we needed to over extrude in order to make it liquid-tight, as we will explain later. We tried to get it even smaller because in theory that would help creating smaller droplets, but when trying to print an outlet with less than 1.2 mm of side, the 3D-printer would print it closed.

Nevertheless, based on its simplicity and the fact that it was working, we decided to move forward with a capillary like based design. In the future we could always roll back and implement a T-junction, but for now we decided to prioritise having a prototype working.

5.2.2 Experimental cycle

After checking that our design was able to generate populations of droplets, the next step was to implement the experimental cycle. This consisted in adding a new input for the aqueous phase, a new input for acetone, and a drain outlet; see Figure 5.5. The objective was to cycle through experiments where the aqueous phase was added into the arena, a population of droplets was generated, and after a minute the contents were removed and the arena was cleaned in order to start a new experiment. From the previous step we already knew that the fittings would not leak when the tubing was connected to a 3D printed device. Thus, our design priority on this iteration was to test if the device would suffer any kind of degradation after cycling experiments through it, because ideally we will want our final device to be running for weeks without human intervention. We also wanted to test if the device could be fully cleaned, and if the dye from the droplets would dye the device itself.

The new device had the same inner channel where the oil was introduced. The acetone and aqueous phases were dispensed from the top of the arena. Also, in one of the bottom corners of the arena there was a draining point which was connected to an outlet. Finally, it can be seen that the surface of the arena was in a slope. It was designed this way so that the liquid would naturally flow to the bottom when drained.

A device with this design was tested for a few days, and we did not notice any apparent degradation. The cleaning cycle seemed to completely remove the contents of the arena, and the dye from the droplets was not dying the device itself.

In the previous design the droplet generator was just a hole in the base of the arena. As said, we had problems where oil would accumulate there instead of making droplets. In order to avoid this, the droplet generator outlet was expanded into the arena, above the base surface, similar to a chimney. This new design brought two new positive things: the area where the oil could accumulate was smaller, therefore

[6]The actual final size was never measured. Each device was slightly different.

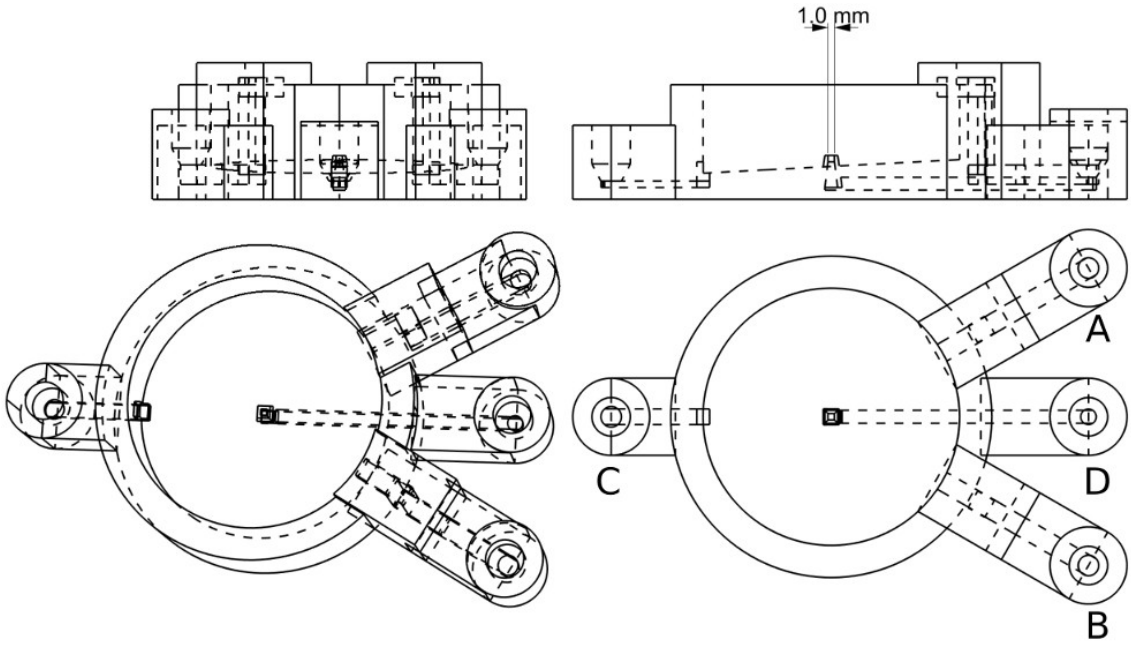


Figure 5.5: Flowbot second prototype. With respect to Figure 5.4, it added two new inlets for the aqueous phase **(A)** and acetone **(B)**, and one new outlet used to drain the arena **(C)**. It used the same channel as before to introduce the oil phase **(D)**.

it was easier to make droplets, and this new design allowed for smaller 3D printed outlets. In this case, the side of the square which acted as outlet was reduced from 1.2 mm to 1.0 mm.

With this chimney-like droplet generator outlet the device was able to generate smaller droplets than before. The smallest ones it could generate consistently had a volume of 1 µl, see Figure 5.6. It is important to note that the liquid pump connected to the oil input used a syringe with a barrel of 500 µl. Independently of the syringe used the plunger could move 100,000 steps with our electronic set-up (see Section 5.4.1), so each plunger step represented 5 nL, which meant that a volume of 1 µl would need 200 steps. We do not know if we couldn't generate droplets smaller than 1 µl because the volume was already too small for our channel size, or because the physical displacement of the plunger was already on the limit of the motor, meaning that moving less than 200 steps did not actually actuate the plunger. We did not test if by using a smaller syringe we could have produced smaller droplets, because 1 µl was already smaller than our desired volume, which was 5 µl, because that is the volume we used with Dropbot (see Chapter 3).

5.2.3 Oil mixer

So far our prototypes only used one channel for the oil input, while our aim was to have four of them in order to replicate Dropbot's functionality (Chapter 3). Similar to how mixing is commonly done in microfluidic devices, we used a serpentine channel in order to mix the different oil phases using internal turbulences, see Figure 5.7.

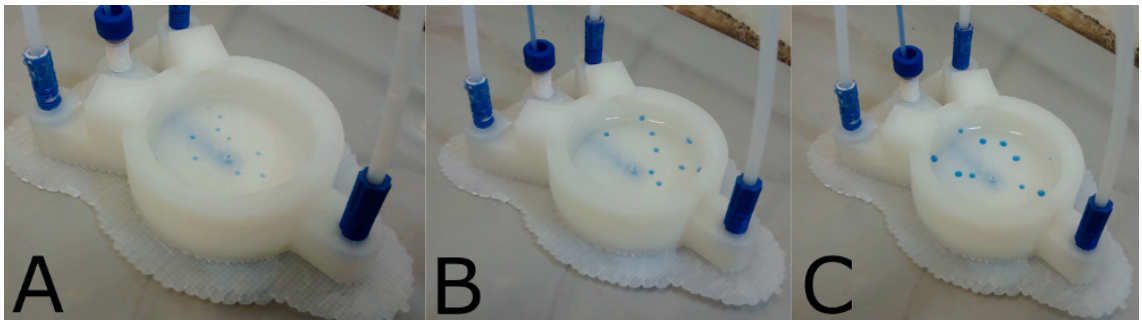


Figure 5.6: Flowbot droplet generation. **A:** 1 µl, **B:** 2.5 µl, **C:** 5 µl.

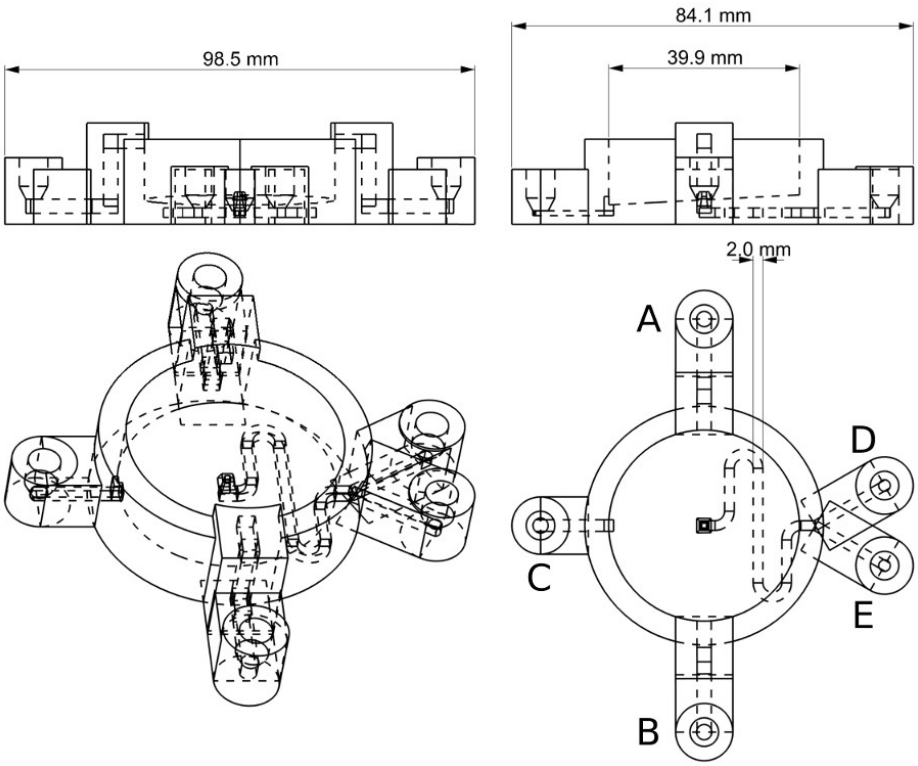


Figure 5.7: Flowbot two phase oil mixer using a serpentine channel. **A:** Acetone. **B:** Aqueous phase. **C:** Drain. **D:** Oil phase 1. **E:** Oil phase 2.

The serpentine channel had an initial square profile of 2 mm side, while the outlet point had a square profile of 1 mm side in order to generate increasing pressure when generating droplets so they would be likely to jump out into the aqueous phase instead of getting attached to the outlet. The first prototype of serpentine channel only had two turns. Also, in order to make space the inputs for acetone and aqueous phase were moved to the sides.

Our main concern was whether it would mix properly or not, because these kinds of designs do not always offer the best results in microfuidic devices because the different phases might go through it in a laminar flow instead of mixing. In our case, we were positive about this because as said our channels were around 50 times bigger than the ones used in microfluidic devices, and this reduced the possibility of laminar flow. By the same reason, the volumes we used were much bigger. The serpentine channel as described here could hold around 500 µl.

In order to test if the mixing procedure was working correctly, two different oil phases dyed red and blue using Sudan dyes were used as inputs. Three different tests were performed. In the first one only one of the phases was enabled. In the second one, the opposite was tested, when only the other phase was enabled. Finally, in the third case, both phases were enabled. If the mixing procedure worked properly, in one case we would get red droplets, in the other blue droplets, and in the final one black droplets. As can be seen in Figure 5.8, the results were as expected, therefore we could assume that the serpentine channel was effectively mixing the oil phases.

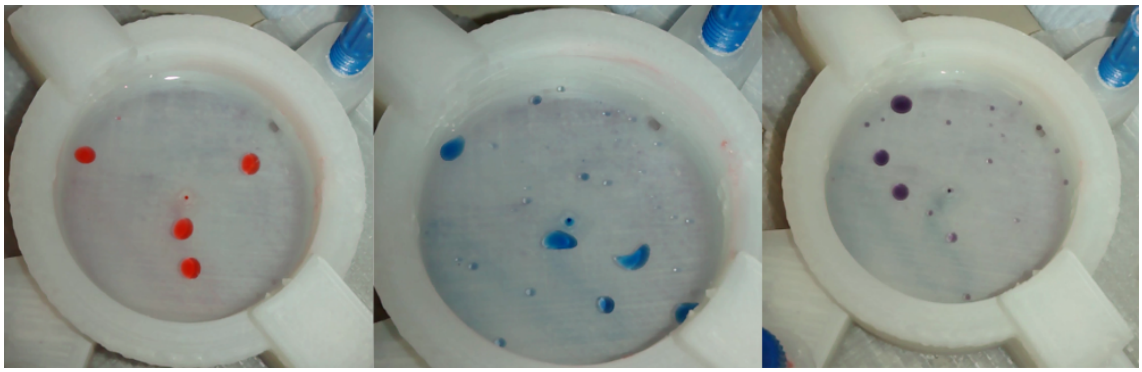


Figure 5.8: Flowbot mixer test. **Left:** Only red phase enabled. **Center:** Only blue phase enabled. **Right:** Both phases activated.

5.2.4 Four way oil mixer – Final design

The prototype shown in Figure 5.9 was the final one, and the one used in all the following experiments, except when said otherwise. Between this prototype and the one described in previous section there were 7 different incremental designs, but we only show here the last one, because the only feature added were the two extra inlets for oil phases. Adding these two new inlets to the previous one, we had a total of four, which is the number of phases we needed in order to reproduce the work done with the previous platform (Dropbot, Chapter 3).

A related and necessary modification in order to allocate the four inlets was the expansion of the serpentine channel. The new mixer went from two to four turns,

with the expectation that the longer path would mix all the oil phases better. The channels that connected the oil inputs to the serpentine had a square profile with 2 mm of side. On the other hand, the start of the serpentine channel where all the inputs met, had a square profile with 3 mm of side. This width was reduced along the serpentine, and at the end of it, where it met the output chimney, it had a side size of 2 mm. While the outlet point, as before, had a width of 1 mm. The side of the profile was decreased through the sequence of channels in order to generate growing pressure that would help to produce droplets, instead of an accumulation of oil phase above the oil outlet. Some other modifications performed in order to obtain this final design consisted of adding more padding of material above, under and to the sides of the inner channel in order to avoid leakage as much as possible. Also, the perimeter of the base of the arena was rounded, and some of the parts of the outer shape were also rounded in order to minimise the quantity of material used when 3D-printed.

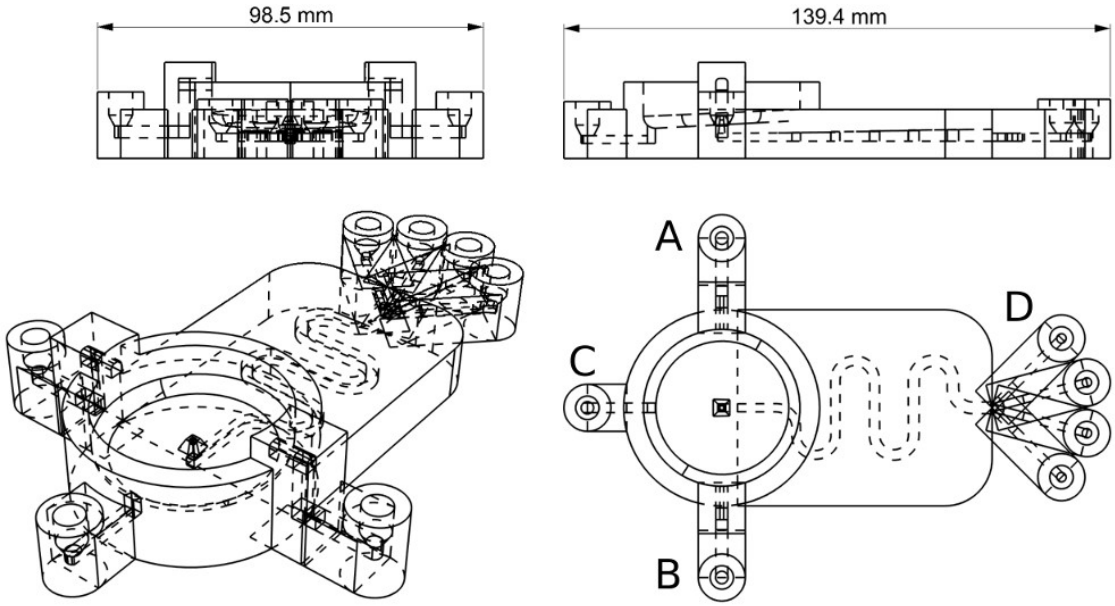


Figure 5.9: Flowbot final design. **A:** Acetone. **B:** Aqueous phase. **C:** Drain. **D:** Oil phases. This prototype was the one used in all the following experiments, therefore it can be considered the final one. From a functionality perspective, its only new feature were the two new oil inputs, which added to the previous one increased the number up to four oil inputs.

The device was initially tested performing some basic experiments, and one of the limitations we found out is that while before, as said, we could produce droplet as small as 1 µl, in this case the bottom limit was 10 µl. Smaller droplets could be generated, but not in the consistent way needed to perform reproducible experiments.

We thought that the main reason for this increase of volume was related to the accuracy of the motor used in the liquid pump, as said before. Using 500 µl syringes, one step in a syringe represented 5 nl. When only using one syringe, only 200 steps were needed to produce 1 µl droplets. But when using four syringes this quantity

had to be divided between four, which meant that each plunger only needed to move 50 steps, and we think this value was under the accuracy of the motor for single shot pulses. We considered the possibility of mixing using the four of them, but only generating droplets using one. This did not work as expected, because the oil phases were connected together, and the disabled ones would absorb some of the volume and pressure from the active one. Based on all this we decided to set the volume of our droplets to 10 µl.

A successfully printed device following this design can be seen on Figure 5.12.

5.2.5 Future design possibilities

Although the design just described is the one we used through all the experiments because it had the specifications we needed, theoretically anything that could be designed using CAD software could be a valid device, and for example we decided to design a similar device with eight inputs instead, see Figure 5.10. A successfully printed device following this design can be seen on Figure 5.13.

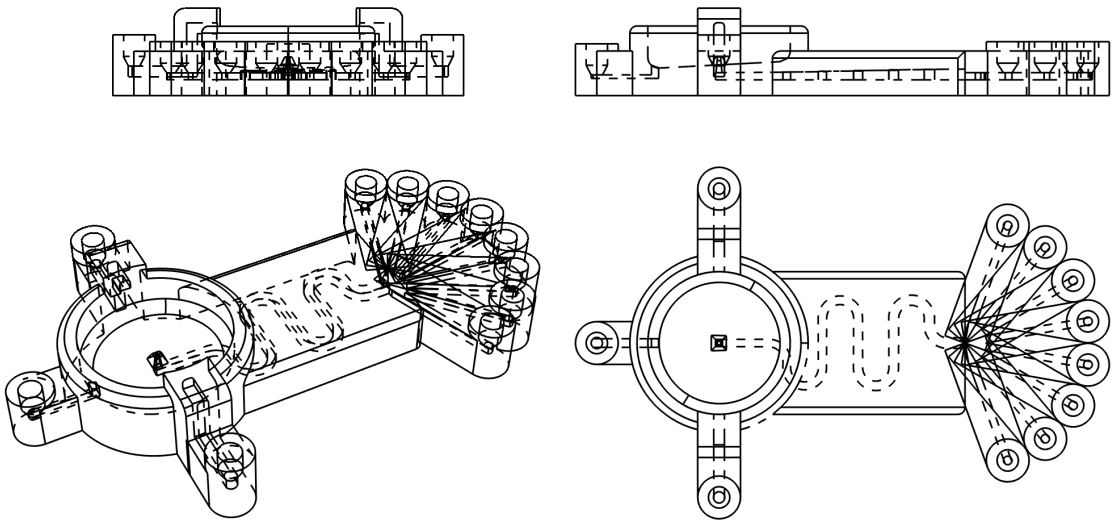


Figure 5.10: Flowbot possible design with 8 inputs. Although a design like this was 3D printed, it was actually never used in any of the experiments here described. See Figure 5.13 for the actual 3D-printed device.

5.3 Manufacturing the device

By manufacturing the device we mean the process that involved taking one of the 3D models from the previous section, and generating a physical device ready to be connected to the liquid pumps. This process consisted mostly in 3D printing the device itself, except the very last step itself, in which the inlets were manually tapped in order to be connected to tube connectors.

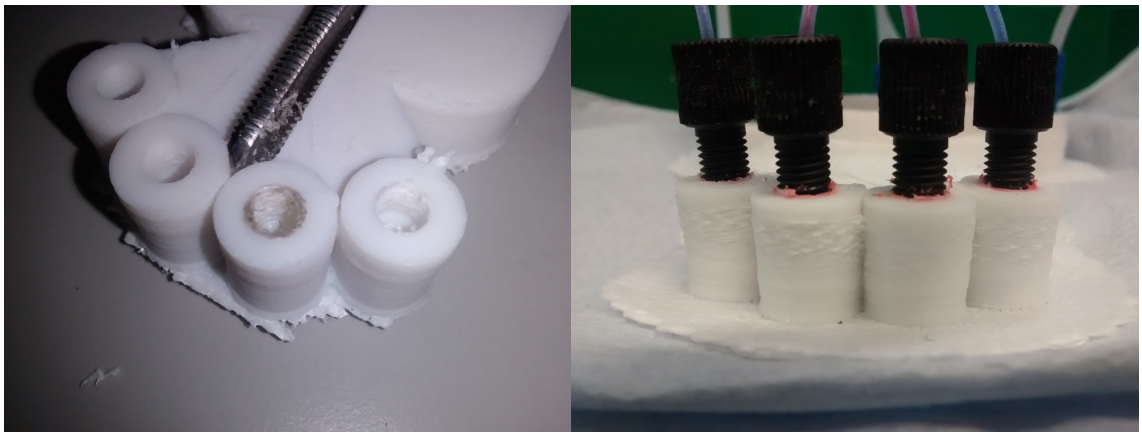


Figure 5.11: Left: A manual tapping tool use to thread the inlets. The size of the thread was "M6". **Right:** Example of a device fully connected without any leakage.

In the experimental chapter it is described how to 3D-print one of the devices (Section 8.2.3, p.186). This protocol is well established within the Cronin group, and it is very similar to the one used to 3D-print the reactionware devices described during the thesis' introduction (Section 1.5.3, p.52).

5.3.1 Preparing the device to be connected to the tube fittings

Once the device had been 3D-printed, the last manufacturing step consisted of preparing its inlets in order to be connected to the tubing system, and consequently to the liquid pumps. In microfluidics devices or similar fluid devices this is one of the most critical steps, because it is the point where leakages are most likely to happen, especially when flowing liquid at high pressure or if there is any back-pressure in the system.

There are many ways to make the connection between tubing and platform. Here, we will use the standard fitting and ferrules that were already used during the Dropbot project (see Chapter 3). The device's inlets were designed to have a diameter of 6 mm, although 3D printers always over-extrude, therefore the 3D-printed diameter was less than that.

The only work to perform was to thread the inlets with a M6 tap tool, see Figure 5.11. It is important to just thread them slightly, otherwise the inlet might become too big. Once the inlets have been tapped, the fitting with the tubing can be inserted. Initially, we screwed it in as much in as we could by hand, and then we used a set of pliers to give it an extra push. In case of leakage in this connection, the easiest possible solution was to use PTFE tape around the fitting to seal it.

5.3.2 3D printed examples

Following the protocol described, we managed to successfully print a lot of different devices. A common question is: What was the success ratio? If we consider all the devices we printed, probably it was around a 10%, if we only consider the last

months when we "mastered" the process, then it was around a 50%. The causes of failure are explained in detail in the Experimental chapter (Section 8.2.4, p.190). Once a device was printed successfully, a few more good ones could be printed in a row until some modification was needed, generally because the warping of the PP sheet was causing problems.

Figure 5.12 shows a successfully 3D printed device, following the design described in Section 5.2.4. Figure 5.13 shows another example of a slightly more complicated device which was printed successfully, although it was never tested. The design of this device is discussed in Section 5.2.5.

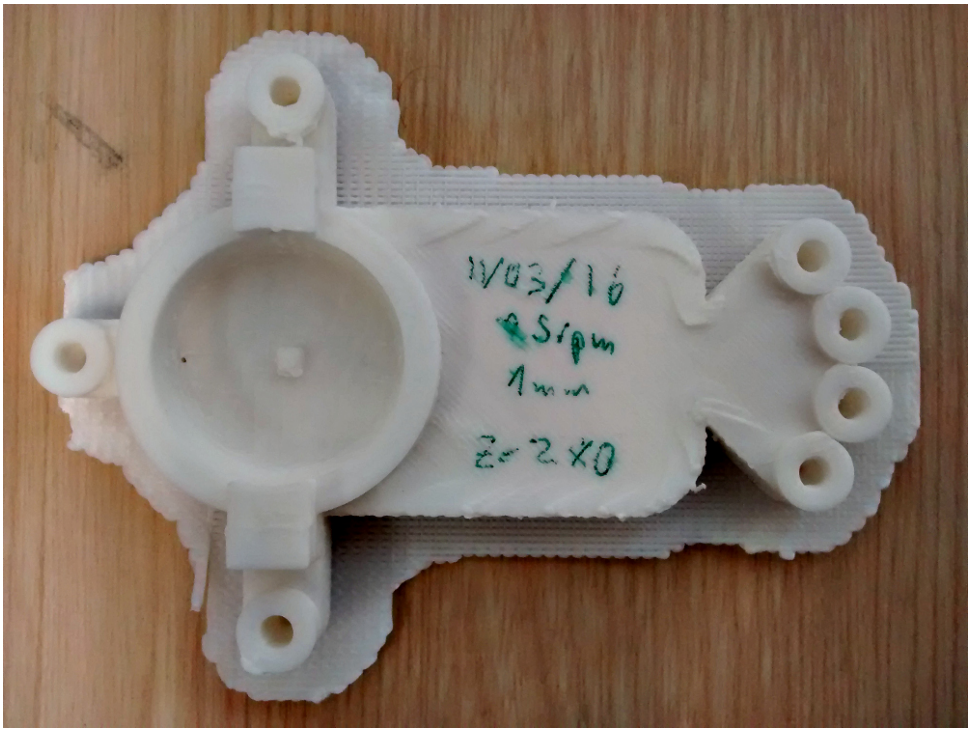


Figure 5.12: Flowbot 3D printed example device.

5.4 Platform set-up

5.4.1 Hardware set-up

After manufacturing a device as described, the next objective was to connect it to a liquid handling system in order to flow liquid through it. The device itself was agnostic to the liquid system used. It could either use syringe pumps, peristaltic pumps or any other technology. In our case, we use modified TriContinent syringe pumps, very similar to the ones described in Chapter 3 (Section 3.2.1.2.1). Therefore, the only work needed was to connect the device to the pumps, and the pumps to some control system. The simplicity of this process was one of its best points, as opposed to the complexity of assembling a full robot like the one described in Chapter 3, because with all the parts ready the platform could be set-up in half day.

Figure 5.13: Flowbot design with 8 inputs. This design was printed only to test if it would be possible. It was never used in real experiments.

We will divide this section in two parts. The first part will describe the connection between the device and the syringe liquid pumps, already partially described in Section 5.3.1. The second part will describe the connection between the liquid pumps and the computer that acted as control system.

5.4.1.1 From the device to the syringe pumps

The device as described in Section 5.2.4 had four inputs for the oil phases, one input for the aqueous phase, one input for the cleaning solvent (acetone), and one output to a waste drum. The seven connections from the device used "Tricontinent C-series" with PEEK 3-way valves. The four pumps used for the oil phases utilised 500 μl syringes, while the other three used 500 ml syringes.

The connection between device and pumps was done using M6 sized PEEK fittings with 1/16" inner diameter and a suitable FEP tubing, and the connection between the pumps and the bulk liquid containers was done using the same fittings but with 1/8" inner diameter, and suitable FEP tubing.

5.4.1.2 From the syringe pumps to the computer

As happened in Chapter 3, the Tricontinent syringe pumps used in this project were not working with its default electronics, therefore we had to design a replacement in order to power them and send orders. Exactly the same electronics used in said chapter would work here, and actually what will be described is just a cleaner version which uses the same components.

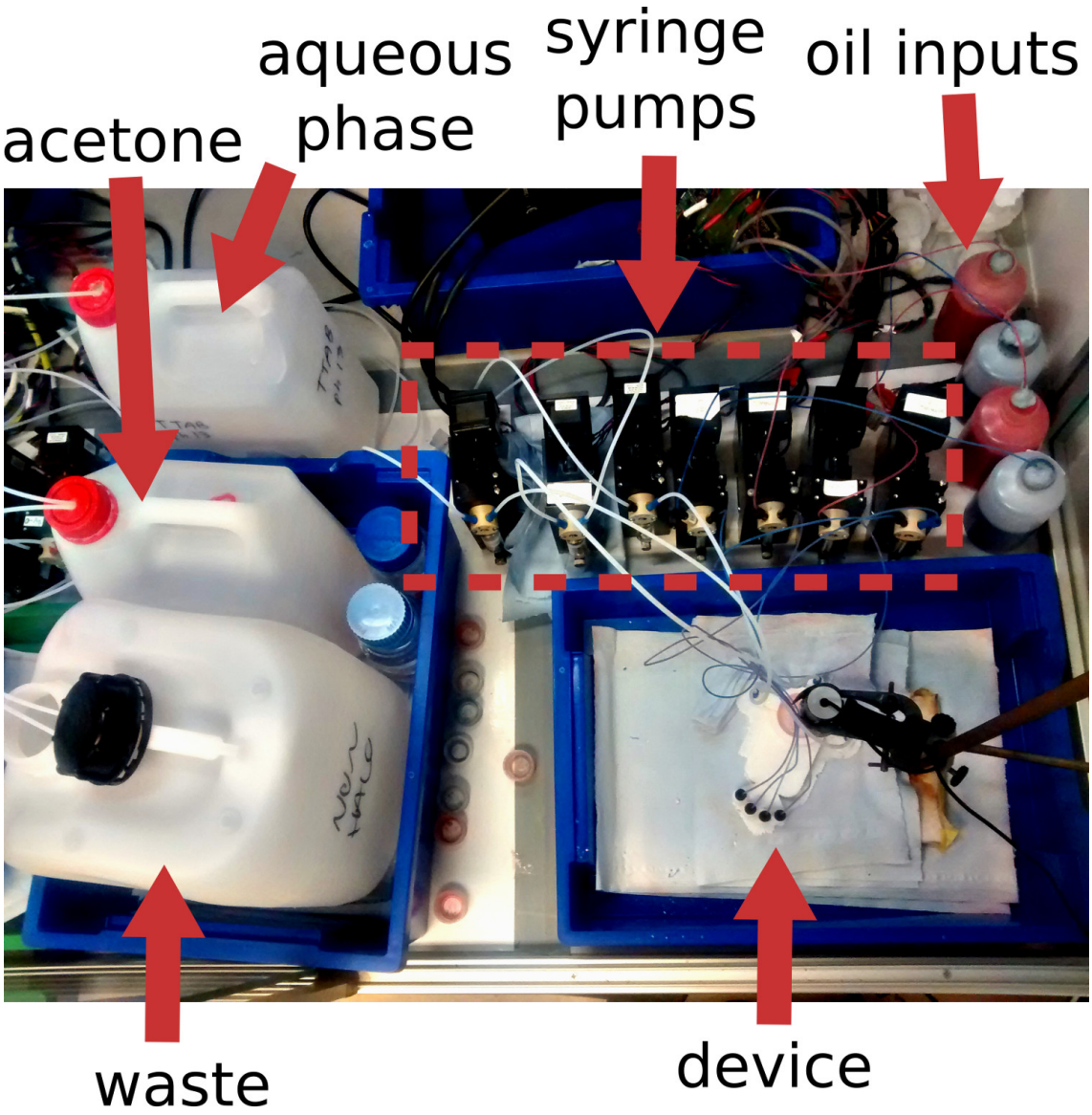


Figure 5.14: Flowbot liquid set-up. The device was connected to seven pumps, and each of these pumps were connected to one of the inputs: four oil inputs, aqueous phase, acetone, or waste output.

The main difference is that we designed a PCB[7] that would be an Arduino shield and that would hold as many A4988 Pololu drivers as an Arduino Mega / Due board is capable of, see Figure 5.15. In our case, because we were using seven different pumps, and each pump had 2 motors, we needed a total of 14 different drivers. The PCB could power up to 22 different drivers. Each of the motors was connected directly to one of the sockets of the PCB, and the Arduino board was connected to the computer using an USB cable.

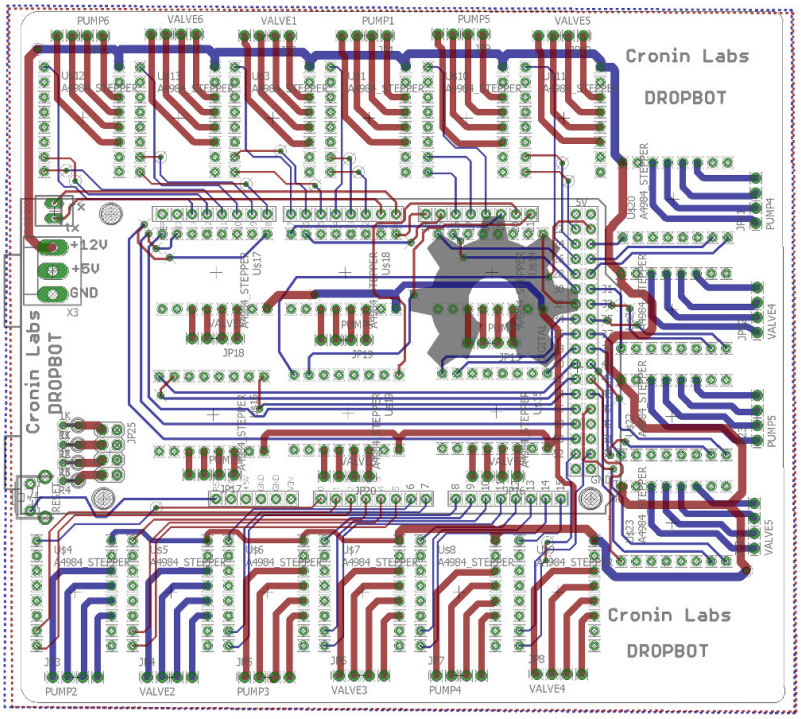


Figure 5.15: Arduino's shield designed to be used with the "faulty" Tricontinent pumps. The pumps will be powered with A4988 Pololu drivers.

While in the Dropbot project we used an Arduino Mega 2560, in this project we used an Arduino Due instead. The reason to do so was that in order to mix the oils using the serpentine channel, it was needed to pump all of them simultaneously at different ratios. Arduino's Mega CPU runs at 16 MHz, and it was lagging when controlling several pumps at the same time[8]. That's why we decided to use an Arduino Due, which runs at 84 MHz, and produced a smoother execution.

5.4.2 Software set-up

We will now describe how the software layer was implemented in order to perform our specific experiments which consisted of generating oil droplets in an aqueous phase, recording the droplets' behaviours with a camera in order to analyse and quantify it, and cleaning the arena in order to be able to perform new experiments.

[7]This PCB was designed using Eagle Cad software by Dr Salah Sharabi as part of this project.

[8]Based on the firmware we wrote to control the PCB. Probably a more optimised firmware can execute the same in the Mega board, but for us it was easier to go with a microcontroller with a higher speed and spend no time trying to optimise the code.

The ultimate target of the project was to optimise oil formulations based on a fitness function, for example maximising the speed of the droplets. Thus, the first objective of the software layer was to generate such formulations in a way that the fitness value could increase through the experimentation. This is a classic Artificial Intelligence problem, and we decided to solve it using a Genetic Algorithm (GA) (Section 1.4.2).

Once a recipe was generated by the GA, the next step consisted of testing it in a real experiment in order to quantify it. In order to test a recipe it was necessary first to prepare the formulation, that is, to mix the oil phases in the ratios specified by the recipe. Then perform the experiments, which consisted in creating populations of droplets in a fixed volume of aqueous phase and recording its behaviour for one minute. Once the experiment was finished, it was needed to quantify it, and in order to do so we used an image processing algorithm that analysed the position of the droplets through the video and returned a numerical value at the end. The last step of the experiment consisted of cleaning everything, so that a new experiment could be executed after it.

Unlike for the robotic system described in Chapter 3, in this project we were only using pumps, so every action required to perform an experiment happened only by actuating them. For example, cleaning the device could mean to add first a fixed quantity of acetone, and then remove its contents. Adding or removing a volume was an action directly related with a defined pump. Each time a pump was actuated, it actually consisted of four actions: absorb with plunger, rotate valve, release from plunger, rotate valve, and each of this four actions were discretised into a single G-code command. Finally, these operations were sent to the Arduino board, which transformed the G-code into electronic pulses which actually moved the physical parts of the liquid pumps.

Thus, the software layer comprised all the work from deciding which experiment to do, to the generation of electronic pulses that would flow the liquid through the device. Between these extremes there were three different domains that could be separated: “high level” operations, “middle level”, and “low level”.

The highest level of operation corresponded to the algorithm that generated the recipes, whether it was a GA or a lattice search or purely random. This level also needed help from an Image Processing algorithm, but also from a database management library in order to save its state between different executions.

The middle level of operation corresponded to all the work from the moment a recipe was given to the generation of G-code commands. In our case a recipe just meant a sequence of numbers that explained how much of each compound was used. For example, considering that we only had four oil inputs, a recipe could be like “20%, 10%, 50%, 20%”. Which meant that oil A used a 20% of the total volume, oil B a 10%, oil C a 50% and oil D a 20%. The total volume was kept fixed, constant quantity though all the experiments. Given a volume needed from one of the liquids, a pump was responsible for flowing it into the device.

The lowest level of operation was the one that took G-code commands and transformed them into electronic pulses.

While the high and mid level ran in a computer, the low level ran in an Arduino board. Figure 5.16 tries to summarise this process in a scheme.

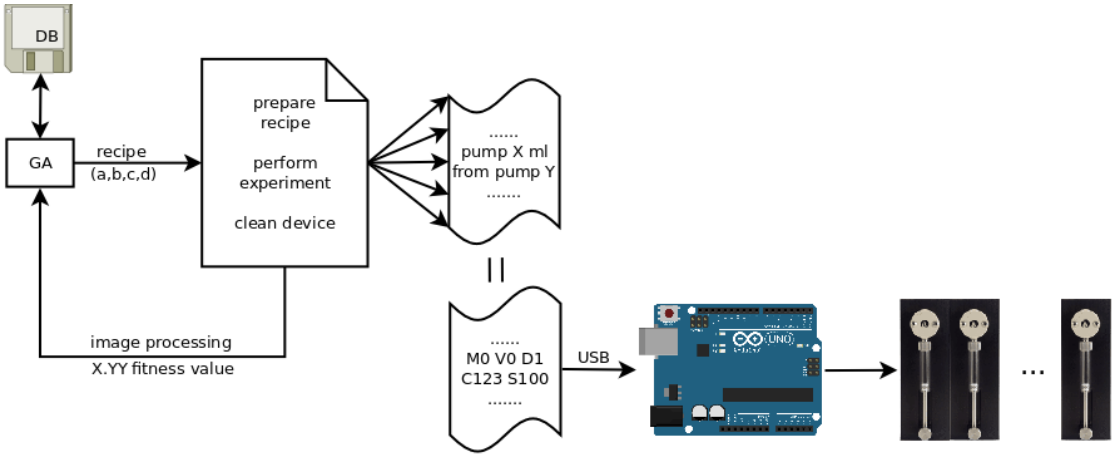


Figure 5.16: Flowbot software layer diagram. The software pipeline started with the Genetic Algorithm, which sent a recipe to Flowbot's main controller. The GA was also connected to a database (DB) layer, which was only used to store and load data, thus the DB was not used in any of the GA calculations. Once Flowbot's main controller received a recipe, in order to perform a full experiment there were several actions that needed to be done. These actions could be divided into: preparation, execution and cleaning. This sequence is very similar to almost any chemistry experiment, and as in chemistry most of the time was actually spent cleaning. Each of these actions related to a sequence of "pump" actions, and each pump act was translated into a G-code command which exactly described how to perform it - mainly it described in which direction the motor would rotate, how many steps it would take and how fast it would rotate. The G-code commands were sent to the Arduino board, which translated them directly into electronic pulses - a PWM signal.

It would have been possible to execute the whole process using only the Arduino board. The reason we decided to split it into the computer and the board was because Arduino's are not that powerful, so it was better to minimise the quantity of work it did. If we have had a "stronger" microcontroller, a neater option would have been to encapsulate everything into it, including the GA, making it completely independent of a central computer.

From a logical point of view this whole process looks very similar to the ones described in Chapter 3, the main difference is that while before all the code was written using Python, in this case we used Labview[9].

We will start by describing the particularities of the GA we used, and then the Image Processing code we used as fitness function. Then we will describe the software used to control the pumps, and finally the firmware used in the Arduino board in order to physically actuate the pumps.

5.4.2.1 Genetic Algorithm

Genetic Algorithms (GA) have already been described in detail during the thesis' introduction (Section 2, p.57). The main difference with the one described during the Dropbot project (Section 3.2.4, p.78) is that the one used here was programmed using Labview[10]. We will not go into detail about the actual implementation, but instead we will focus here on the differences to the GA used before. Some of these differences were caused by the use of Labview, while some others were caused by the fact that the hardware and experiments done were a bit different.

The GA was programmed using LabView 2015 standard libraries. Each GA run, except when described otherwise, consisted of 10 generations, and each generation had a population of 20 individuals. Each individual was defined by its recipe, which was the ratio between the four oils used (1-octanol, DEP, octanoic acid and 1-pentanol). Each of the oils was assigned a real number between 0 and 1, and the sum of the four oils for a given recipe was always 1. The first generation was constructed by assigning random generated recipes to each individual. Each individual was tested five times, and the average of these five executions was returned as its fitness value. Once all the individuals from a generation were given a fitness value, a new generation was constructed by choosing 10 parents from the just finished generation using the roulette wheel algorithm, where the individuals were ranked using their fitness value (the higher the better). A given parent could only appear once in the following generation. The other 10 individuals were constructed by crossing the parents in pairs, applying a random position one point crossover, and a gaussian 10% mutation. The final recipe was then normalized to 1.

In the previous GA implementation each of the genes was represented using a binary notation. In this implementation each gene was represented with a real number instead, as this was simpler to code in Labview.

[9]The main reason we used Labview is because we wanted to learn it. Our degree of expertise with Labview was lower than with Python, which meant some of the functionality could not be fully ported. If we had to do this project again, we will probably use Python (or a similar programming language) instead.

[10]It was coded by Dr Mike Lee as part of another project, and reused here.

Each generation consisted of a set of parents and their offsprings. While before given a new generation we only tested the offsprings and carried the value of the parents, in this new implementation we tested each individual, meaning that the parents were tested again every time they appeared in a new generation. The objective to do so was to also test the reproducibility of a recipe, because a "good" parent had to return high values consistently in order to be kept in the mating pool.

5.4.2.2 Lattice search

The other mechanism we used to generate recipes was a lattice search. In the previous chapter a lattice search was initially used a stress test mechanism just after the robot was built because we wanted to check if it would be able to endure continuous work. The results were very interesting, and that is why we also decided to repeat it during this project. It was also programmed using Labview 2015. See the experimental chapter for a full description (p. 195, Section 8.2.6).

5.4.2.3 Image Processing

An experiment started once a population of droplets had been placed in the aqueous phase contained in the experimental arena. From that point on and for one minute the experiment was left unattended while a camera recorded everything that happened. The objective of the image processing library was to analyse the video and quantify its behaviour, returning a real number associated with a predefined characteristic.

In Chapter 3, the three characteristics we studied were movement, division and vibration. Herein, we will only focused on one of them because the focus of this project was instead on environmental changes, as described during this chapter's introduction. The analysis of the video itself the same as before, and the difference resided on how the data generated by this analysis was processed. Therefore, we can say that this process could be divided into two steps: (1) given a video produce numerical data (2) transform the data into a fitness value based on a defined characteristic. In our case, the numerical data represented the position of the droplets on each frame. Thus, we had a list that had as many elements as frames in the video, and where every list element contained information about the number of droplets in that frame and their position.

It was our objective to implement the image analysis using Labview, because it is very good with computer vision and image processing, but we did not have the time to do it, which meant that we re-used the Python library explained during the Dropbot project (Section 3.2.3, 72). The previous image processing code could not run live while a experiment was happening because it took more time to process a frame than what is available at 30 frames per second (FPS)[11]. What we did before was to only save the video while the experiment was happening, and then while the robot was cleaning the Petri we would also analyse the video at the same time and return the fitness value. Programming something like that in Python is simple, and although Labview makes it very easy to program concurrent code, we did not know

[11] At 30 FPS the camera would generate a frame every 0.0333 seconds. This means that all the process from the image processing must take less than that

how to call the Python library in a parallel way. Therefore, this time we needed the image processing to happen live.

In order to do so we simplified the code used in the previous project. Before, our image processing consisted of two different algorithms that were merged at the end: one processed the images based around a watershed algorithm, and the other processed them using a background subtraction.

We decided to only use the background subtraction algorithm. This algorithm works very well for fast moving droplets, but it fails when tracking stationary objects. Thus, our algorithm by default would focus on droplets that would move, and would disregard the ones that did not. Therefore, if a droplet was not moving it could be said that is "dead" or out of energy.

The other major simplification with respect to the code used before is that previously we used to do a "Hough transform" to detect the Petri dish, while now we removed this step and just set a constant area through all the experiments that would be considered the experimental arena.

Finally, the fitness function used here is what we called before "division", which was implemented as counting the number of droplets "active" at the end of the experiment. Therefore, on one hand we wanted the droplets to actually divide in order to maximise this number, but because our image processing was based solely on a background subtraction, we also wanted them to move. So we can say that the new fitness function counted the number of moving droplets at the end of the experiment.

A positive side effect of the new approach is that only performing a background subtraction was actually the right thing to do if we consider that here we had different environments that added different obstacles (see Section 5.5 for a description of these structures) into the field of view. As said, our previous image processing algorithm had two different techniques merged together: background subtraction and morphological operations which aimed to find closed structures. It is very possible that the morphological operations would return as "droplets" the obstacles in the environment. We could probably modify the code to work around this, but only using a background subtraction nullified completely this problem.

A final note about the background subtraction is that the backgrounds database, that is, the information the algorithm builds up to discern between background and foreground, was restarted at the start of every experiment. This meant that (a) during the first two or three first seconds of every experiment the results were very noisy because the algorithm did not know yet what was background and what was foreground, and (b) we had to set-up a very small memory so that the algorithm returned good results early on. This had the drawback that if a droplet stopped moving after a few seconds it was very likely be considered part of the background until it moves again. We worked around this by saying that the algorithm only looked for "active" droplets, and that if a droplet did not move for a period of time, it was considered "inactive", so it was good that the algorithm forgot about it. The reality is that we would have liked to save the backgrounds database between experiments, but this functionality is not implemented yet in the computer vision library we were using (OpenCV).

See the experimental chapter (Section 8.2.12, p.198) for a detailed description of

the implementation.

5.4.2.4 Middleware[12]

The objective of the middleware layer was to receive an experiment recipe, which as said was just sequence of numbers determining the ratio used in the formulation, like [.2, .35, .15, .3], and generate a sequence of G-Code commands which exactly described the hardware operations the platform needed to perform, like for example "rotate motor 7 clockwise direction for 30 steps".

Although the translation from recipe to G-code commands could probably be performed in a single software step, we decided to divide it into several steps because that made it easier to read, test and debug. All the code implemented in this section was written using Labview 2015.

It might be interesting to read first the experimental protocol (Chapter 8 p.196, Section 8.2.9), which details exactly how the experiments were executed, because the aim of the middlware is to actually implement the protocol described. Every experiment followed the same protocol, therefore all of them had exactly the same software implementation, and the only difference was the exact quantity used for each oil. This means that instead of generating from scratch every experiment, we could have generated one of them and all its related G-code commands, and then inject the mixing values when required. It was not done this way because the computer or CPU time cost of the middleware was very small when compared to other parts of the software, like the image processing, or when compared to the actual time cost of moving and actuating the physical parts of the platform. Thus, although a lot of the "software" work could have been optimised, it was better to keep it this way for simplicity.

The first step of the middlware was to describe the experimental protocol from a high level point of view, independent to the actual hardware implementation. This description would be the same when executed in a robot, in a fluidic platform or when performed by an actual researcher. The main steps of the high level implementation divided it into: mixing, cleaning mixture, performing experiment, and cleaning experiment. As said in the protocol section, each mixture was tested five times, so the last two steps were iterated five times.

The next step was to associate these "high level" actions with the pumps. We used seven pumps, and each pump was responsible for only one input or output. From one to seven our pumps were: (1) aqueous phase, (2) 1-octanol, (3) acetone, (4) arena output, (5) DEP, (6) octanoic acid and (7) 1-pentanol. As an example, a cleaning cycle might need to first add acetone to the arena, and then remove all the contents. In order to add acetone we would actuate pump three, and in order to remove its contents we would actuate pump four. Each pump action was actually defined by four numbers: pump id (as said), speed plunger down, speed plunger up

[12]In the context of software engineering "middleware" usually refers to a software layer that acts as a "glue" between different software layers - like between an operating system and a database. Although by the technical definition the software described in this section is not really a "middleware", we used that name through the development because its only objective was to glue the genetic algorithm library with the actual hardware platform.

and number of steps. The maximum number of steps was 100 000, so for example 50 000 would mean the plunger would travel half the distance.

The last step took each of these pump actions and transformed them into G-Code commands which followed the syntax described in Section 5.4.2.5 (Firmware). Each pump used two motors: one to actuate the syringe moving the plunger up or down, and one to actuate the valve. Therefore each pump action, like adding 1 ml of aqueous phase into the arena, needed four motor actions: move the plunger down to absorb the liquid, rotate valve to output, move the plunger up to release the liquid and rotate valve to input. Each of these motor actions was described exactly by a G-Code command which defined which motor to actuate, in which direction and for how long. The G-Code commands were the information sent using Labview's VISA[13] library to the Arduino board through a USB cable.

The very last thing the middleware was responsible for, once the experiment was finished, was to return to the GA the fitness value of the experiment. A recipe was repeated five times, so the fitness value returned was the average value of these five repetitions.

See the experimental chapter (Section 8.2.13, p.198) for a detailed description of the implementation.

5.4.2.5 Firmware

The last step of the software part consisted in given a G-code command, generating the sequence of electronic pulses that would actuate the motors. Each G-command only actuated a single motor, and all the motors used were stepper motors. So, a G-command would identify a motor, and say in which direction it should rotate, how many steps it should rotate, and how much time it should wait between steps. This wait time between steps is in the order of a few nanoseconds, and the lower this value is, the faster it will rotate.

More precisely, our G-code commands were structured as follows:

$$\mathrm{P}\alpha\ \mathrm{M}\beta\ \mathrm{D}\gamma\ \mathrm{C}\delta\ \mathrm{S}\eta\ \mathrm{E}\theta$$

Where:

- P represents pump, and α is a value between 0 and 6 that identifies the pump.
- M represents motor, and β is either 0 or 1, depending if the motor enabled is the plunger motor or the valve motor.
- D represents direction, and γ is either 0 or 1, depending if the motor is rotating clockwise or counter clockwise.
- C represents code, and δ is an integer number that the firmware would return back to the computer once the action is finished.

[13] Virtual Instrument Software Architecture.

- S represents speed, and η is an integer number that represents the waiting time between steps, in nanoseconds.
- E represents steps, and θ is an integer number that represents the number of steps the motor rotates.

The only difference with the G-code syntax used in the previous project (Chapter 3 Section 3.2.2.2), is that in this case there was a new parameter C, which represented a code that identified a command sent. Once the command had been fully executed, this code was returned back to the *middleware* described above. This new parameter is required now because this firmware was able to execute several actions at the same time, as opposed to previously when the commands were executed one by one. The firmware had to be synchronized with the middleware, and assigning "ids" to actions was an easy way to know which ones were sent and were unfinished, and which actions had just been finished.

The reason why we needed to execute several pump actions at the same time was because: (a) our mixing process was based on a flow mixer where the different reactants were pumped in at different ratios,[14] and (b) some of the pump actions could be actuated simultaneously in order to save time.

Because we needed the new firmware to actuate different motors simultaneously without blocking the execution - so halfway through an execution it could receive new commands and also execute them- the new firmware was rewritten from scratch. As before, it was based in the Arduino architecture and written using their C language implementation.

We based the new implementation on a Finite State Machine (FSM) architecture. Each motor represented a state[15], and each state contained information about the number of steps left to execute[16]. Initially all of them had zero steps left to execute. The FSM was continuously looping through all the states checking if there was any work available. At first they were all zero, so nothing was done. Once a command was received stating that a given motor must execute X steps, the information was updated in the state represented by that motor. Meanwhile the FSM was looping checking if there was any work available, and once it arrived to the updated state, it would execute one step and decrease the number of steps available by one. Some new commands might arrive stating that other motors must actuate, and their information was update in their defined states. The FSM was looping through all of them, executing steps when available, and decreasing their counters.

This implementation worked as expected, but the main problem was that the Arduino Mega was not fast enough to execute the steps at the required speed,[17] and

[14]This is different to what happened in Dropbot, when the mixer was done in batch, and therefore the different reactants could be added to the mixture at different times.

[15]There were seven pumps, and each pump had two motors, therefore there were a total of 14 motors or states.

[16]They contained more information, but simplifying it to the number of steps is enough to explain how it was implemented. The firmware C file is attached with this thesis in case the reader wants to check the actual implementation.

[17]For example, a common speed used was 30 ns of wait between steps. If the Arduino Mega were doing nothing more and had a 0 cost execution per iteration, that would need a CPU clocking around 33MHz, while the Mega ran at 16MHz.

this is why the board was swapped for an Arduino Due.

5.5 Generating different environments

As said during the introduction of this chapter, one of the most interesting features about 3D printing the whole device in a monolithic piece is that we could change the structure of the part that acted as an arena.

Initially, as shown on Figures 5.9 and 5.12, the arena was an empty environment, similar to the Petri dish used during the previous research (Chapter 3).

Our first custom arena consisted of a series of pillar-like structures. These pillars had a diameter of 2 mm, and a height just enough to raise above water level. The arena was densely populated with these pillars in a linear / grid way, because we wanted to force their interaction with the oil droplets (Figure 5.17) The initial results were interesting (Section 6.4.2), so we decided to move forward with it and to test it against our battery of experiments.

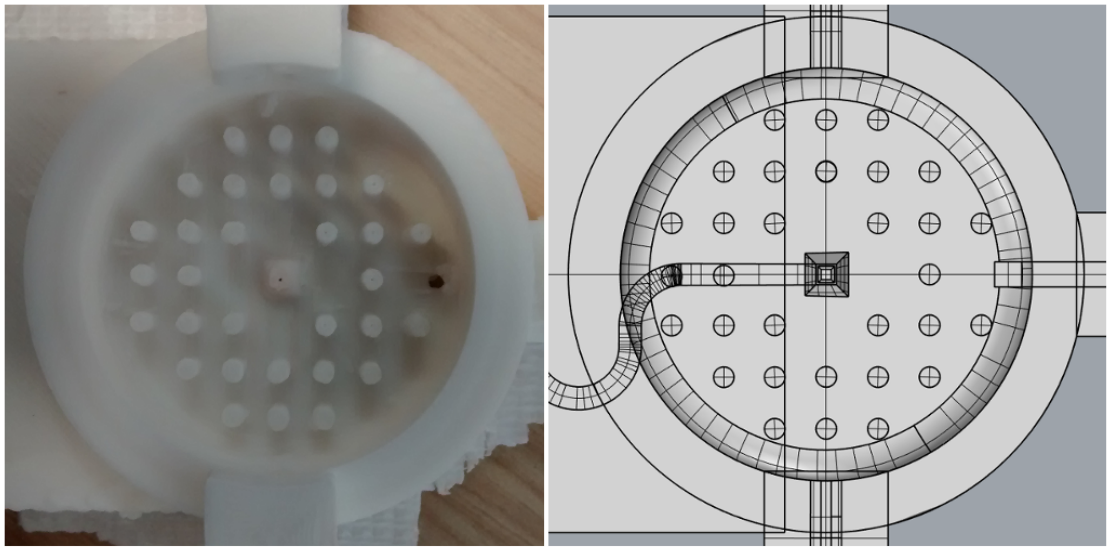


Figure 5.17: Flowbot pillars arena. This arena was manually designed. The objective was to create a densely populated arena that would force interaction with the oil droplets.

This environment was manually designed by us. We decided about the shape of the structures and their positions. We had no prior knowledge about what could happen, so as said we simply tried to force interactions by densely populating it. There was no reasoning behind its design apart from making something simple and 3D-printable.

In order to move forward we decided to let an algorithm design the environments instead. There were many different alternatives that could be applied here, but we decided to use a Lindenmayer system (L-system, [Lindenmayer, 1968]). The main reason why we chose an L-system was because they had already been used in similar problems where natural patterns and plant developments were generated in a procedural way. They are easy to design, the rules that govern its development

are easy to modify, and they allow for the incorporation of stochastic factors that would potentially generate a range of different environments.

We had two options about how to use the L-system to design new environments: (a) we could let the L-system design the unit structure to be repeated in a pattern or (b) we could use a circular pillar as described as the base structure and let the L-system design the pattern instead. Using a Python script and the "turtle" Python library ([Python-Turtle, 2016]) in order to draw the results, we explored both options.

When letting the L-system design the base structure, we used as a basic unit a square with 0.1 mm of side. Using random based conditions for each step, the structure could grow in four different directions, or could leave an empty square. Figure 5.18 shows the different structures generated.

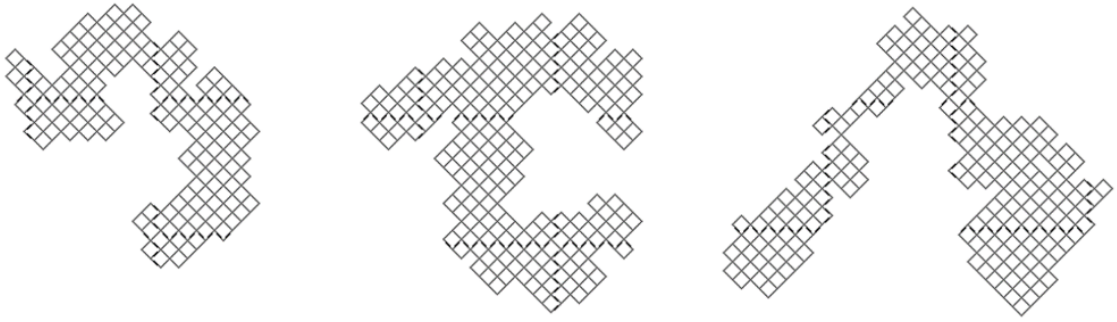


Figure 5.18: L-system generation of environmental base structures. Each square represents 0.1 mm, and they were generated stochastically. Although many of these structures were generated, none of them seemed to be really interesting.

When using a pillar as a base structure, and letting the L-system generate the pattern, we set manually an angle at which the next pillar could be generated from the previous one, and we used a random number generator to decide if it would place a pillar or jump the distance instead. Figure 5.19 shows the different patterns generated.

We decided to move forward with the second option because we thought that the cave-like structures that were generated would have an interesting interaction with the droplets. Using CAD software, an environment was created manually based on the patterns generated before, and then a device was 3D printed successfully. See Figure 5.20. Devices exactly like this were the ones we tested and the results can be seen in Section 6.4.3.

We only had time to test one of these designs. We would have liked to test more of them, and also to test the ones where the L-system generated the structure itself, but as with many other things we did not have time to do so. Also, as just said, the environment was generated using the "turtle" library from Python to draw them, and then copied manually into a CAD software. Our initial idea was to draw them directly into the 3D design, using for example "OpenSCAD" ([OpenSCAD, 2016]), a program where you code the 3D models in a C-like programming language. This way, it could be possible to code the L-system in OpenSCAD and let it generate the environments in the arena. We did not have the time to do, but it is one of the many things left to do in the future if we ever go back to this project.

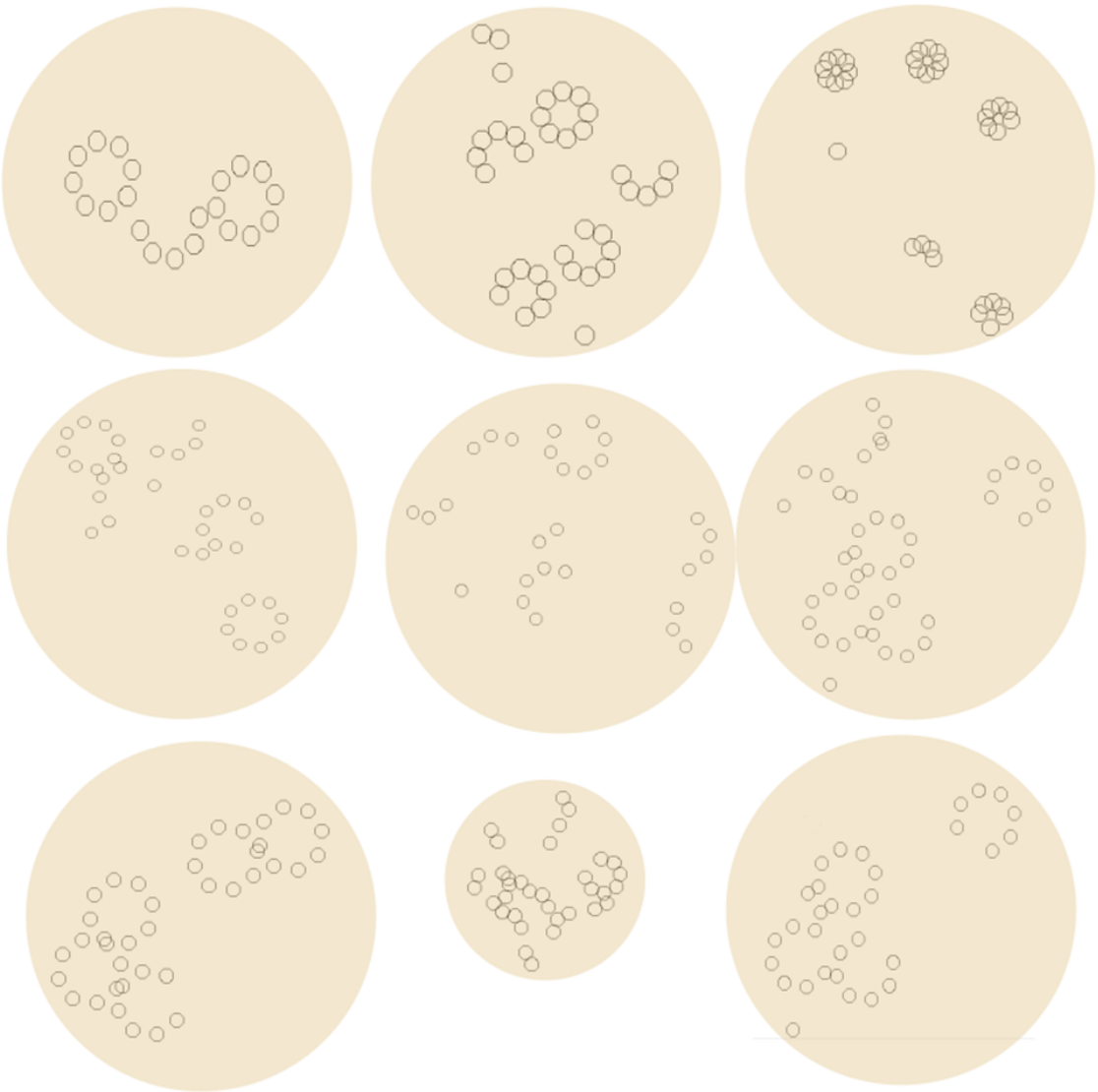


Figure 5.19: L-system generation of environmental patterns using a pillar as base structure. The beige background is not supposed to be the Petri dish, but it is just used as a mean to group them into different environments. Although the environments had different areas, all the pillars were the same size, and the environment area only depended on how spread the pattern was.

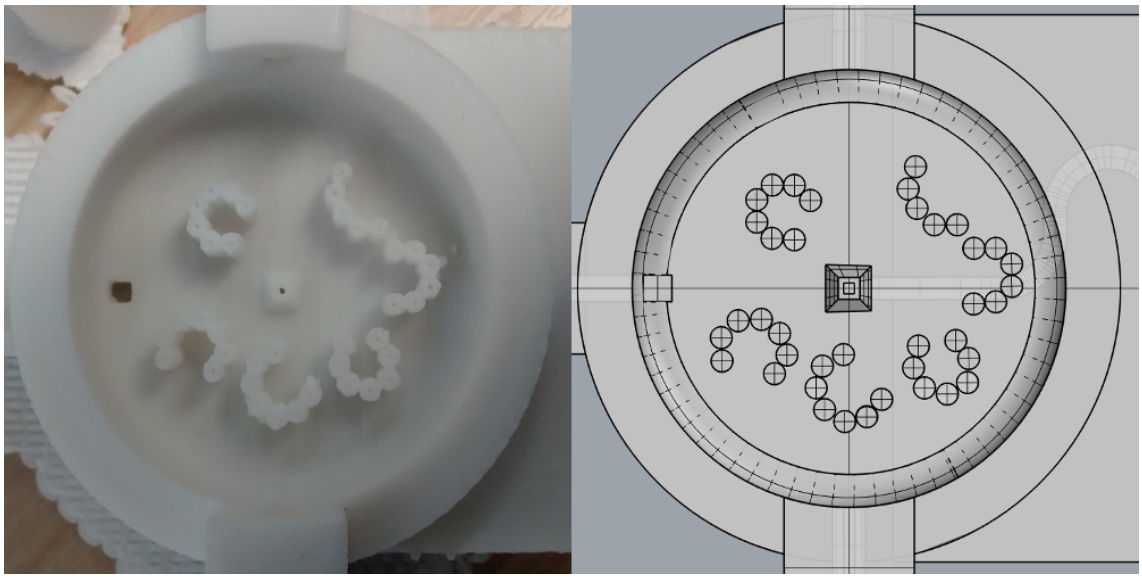


Figure 5.20: Flowbot with "caves" arena. This arena was designed based on the patterns generated by the L-system, in particular we wanted to study how the droplets would interact with the cave-like structures.

5.6 Discussion and future work[18]

Designing and building the platform took roughly a year of the PhD, and it can be said that we were somewhat successful, because we got the platform working and producing results as we will show in the next chapter. Nevertheless, there are a few things that would be needed to be addressed if we were to start this project again:

- The devices manufacturing process was full of problems. Even though we almost always followed the same protocol to manufacture the devices, most of them were actually not working - like failed prints or devices that leaked. Changing the protocol was not an option because we were using a commercial FDM 3D-printer, and trying to modify it would add more complexity to this project. We also played with all the configuration parameters as much as possible, and we think we tuned the process as much as we could. The problem resided in the polymer used, PP. As we have explained before, it was a very difficult polymer to print with. A first possibility would be to use another one, although none of the common polymers used for 3D-printing are good for chemistry. We tried FEP (fluorinated ethylene propylene) which also has a very high chemical resistance, but it was nearly impossible to extrude. Another possibility would be to use another 3D printer technology instead of FDM, but as far as we know no other method uses a material with such high chemical resistivity. We could also use an external company to manufacture the devices using molds or similar, but that would make impossible to make the internal

[18]Because this chapter only covered the platform itself, and what can be considered a "method and materials" section from a paper, a more complete "discussion and future work" section about this project can be found on the next chapter (Section 6.7).

channels. Thus, as far as we know, our method was the only available, but it would be very interesting to try something different.

- Once we had a device working, the main problem we had was creating droplets in a consistent way. What happened more often than we would have liked is that the oil would accumulate in the outlet forming a heap instead of extruding and forming droplets. We overcame this by producing bigger droplets and using faster pulses, but this had the drawback of not being able to produce smaller droplets. The first option if we had do this again would be to try a T-junction like droplet generator, instead of the capillary one.

- As for the software, it was a mistake developing everything using Labview. If we had to go back, we would use a more standard programming language.

- The environments were generated using an L-system. We really did not have time to explore more options. It would be very interesting to compare it with different procedural generator algorithms.

- The environments generated with the L-system were at first created as a picture, and then we manually replicated them using a CAD software. It would had been more interesting to encapsulate all the process in the CAD software instead, so that it does not require human intervention.

- The environments were 3D printed in the device in a single step. It would had been helpful to print them as inserts instead, so that we always used the same device, and we only changed the insert. This way, we would not had need a new device every time we needed a new environment. This would have minimized some of the manufacturing problems. This possibility was discussed, but never implemented.

There are also some things that would be interesting to implement as future work:

- So far, once an environment is generated, we needed to print a device with it. This meant that we had to use the same environment for the whole GA run, because it was something we could not modify dynamically. It would be very interesting to use some technology that would let us change how an environment is dynamically.

- The platform as described used syringes pumps which were connected to an Arduino board, which was connected to a computer. From an industrial perspective it would be interesting to create integrated devices that do not need a computer, with smaller, cheaper pumps.

- It would be also interesting to try a similar project using microfluidic devices instead.

6

Flowbot results

This chapter assumes that a working platform has already been built, and focuses instead on the results obtained.

It will start with a brief introduction (Section 6.1), which only aims to complement the one already given in the previous chapter, then it will describe chemistry used (Section 6.2), and the platform itself (Section 6.3).

It will follow with the main results obtained, in particular the ones related to the use of evolutionary algorithms (Section 6.4) and a lattice search (Section 6.5), followed by a analysis of the results (Section 6.6), and finishing with the discussion and future work (Section 6.7).

6.1 Introduction

The introduction of this project, its motivation and its main goals were already described in the previous chapter and during the thesis introduction. While the previous chapter focused on how to build the platform, this chapter will focus instead on the experiments it did and the results obtained.

Once the platform was fully working from a mechanical point of view, and once we could send it recipes to test, our next step was to perform a series of experiments in order to check if it could drive an evolutionary process like it was done during the Dropbot project, and more importantly, to check if by using different arena environments we could obtain different results.

A necessary step before any work was done, though, was to verify that the platform could run continuously and unattended for long periods of time. During Chapter 3 we did so by running a lattice search. This time we decided to go directly for a genetic algorithm (GA) execution, in order to test its robustness and validate its results. A lattice search is a good method to check if the platform does work from a mechanical point of view, but it does not say much about the "quality" of the experiments, which can be measured by "signal to noise" ratio. When using Dropbot we were not concerned about the possible noise or contamination present in the system because the protocol was very similar to how experiments are performed manually, so we expected it to be negligible. On the other hand, because the mixing protocol using Flowbot was novel and untested[1], we were concerned about the quality of the mixture and the possible system contamination. That is why we decided to test the device directly using GA runs to test both robustness and noise.

[1]Compared to a beaker with a magnetic stirrer, which is the standard mixing procedure in chemistry.

In any platform, even industrial ones, there is always noise associated with the results. The noise can be generated by several different sources, like an imperfect calibration, some of the components getting old, unexpected leakage or contamination from previous experiments. In good industrial platforms the proportion between noise and signal is minimal. In our case, we did not know how this proportion was represented, but we based ourselves on the premise that if the platform could undergo a GA, and the fitness value increased through generations, the platform had some degree of reproducibility[2], and the signal component was bigger than the noise.

Based on our platform design we considered that the noise could be sourced from:

- Contamination from previous experiments, i.e. the platform was not perfectly cleaned between experiments. If this was a problem, it would mean that every experiment is contaminated by a window of previous experiments.
- Internal leakages we could not detect. Very related to the previous point. If internal leakages were a problem this would mean that the results would go to an average. Thus, after "n" experiments, no matter what the input was, the platform would always output very similar values.
- Serpentine mixer not working as expected. The main problem we could expect was laminar flow.

Because our objective at this point of the project was to produce "research" results and not "industrial" ones, we did not stress test our platform for any of the previous points. We considered that if it could run an evolutionary algorithm, and growth was apparent, we could deem that whatever the SNR was, it was good enough for us. Based on this, we would be very careful to use this platform with other experiments, because cautious testing might be required.

Once we could verify that the platform worked as expected and it showed growth in the evolutionary trajectories, our aim was to prove that environmental changes had an effect on the results, and that was the main motivation of our experiments and what will be described in this chapter.

Generating all the data took approximately one year. Our final database consisted of around 20,000 videos, which used about 10 TB of data, and that only contained the "good" experiments we saved, because many more were produced that were discarded for several reasons. Once we had two platforms running at the same time, on a good day we could generate around 200 videos.

[2]There are many ways to check the signal-to-noise (SNR) ratio of a platfom. An easy way to check it in a platform like ours would be to iterate through a series of experiments. Every iteration should execute the exact same experiments but in different order. While this is happening, the data is collected. A perfect platform would output the same data in the first iteration or in the last one. Comparing the deviation of the results, the SNR can be obtained. Another way to do it would require to repeat only one experiment continuously, and comparing its output through iterations. This technique would measure what is described as "white noise" in electronics. The problem with our system is that in chemistry contamination is a much bigger problem than systematic noise.

6.1.1 Chapter objectives

- Run a genetic algorithm through the device with an empty arena and show that the fitness values increase.
- Run a genetic algorithm through the device with the "pillars" arena, and show again that the fitness values can increase.
- Run a genetic algorithm through the device with the "caves" arena, and show again that the fitness values can increase.
- Perform a hybrid GA run, where the first 10 generations use one of the arenas, and the following 10 generations use another of the arenas.
- Perform several different variations of the hybrid GA run.
- Perform a triple hybrid GA run where the three different environments are tested in a consecutive way. Thus, the first 10 generations can use device A, the following 10 device B, and the last 10 device C.
- Perform a lattice search for each of the devices.
- Show that the environments made a difference. This is our null hypothesis. It might be the case that there is no difference at all.

6.2 Oil droplet system

Five components were chosen for the droplet system: 1-octanol, diethyl phthalate (DEP), 1-pentanol and octanoic acid in the oil phase, and tetradecyltrimethylammonium bromide (TTAB) at pH 13 in the aqueous phase, with the purpose of covering a wide range of divergent polarities, densities, viscosities, solubility, and different possible interactions that may occur at the interface. The objective was to create droplets that would provide a chemical compartment related to the aforementioned attributes, aiming for their stability and the ability to move and divide, see Figure 6.1.

Previous tests (Chapter 3) showed that 1-octanol oil droplets underwent rapid movement and division when placed in the aqueous phase although 1-octanol is not deprotonated at this pH. Droplets of 1-pentanol are less hydrophobic, resulting in a very fast dispersion once they were placed in the aqueous phase. On the other hand, DEP has very low solubility and it is denser than water, resulting in very stable droplets that would sink and remain inactive. Finally, octanoic acid is very hydrophobic and would create stable moving droplets, but it is also deprotonated at high pH forming an anionic surfactant.

One of the reasons for focusing on chemical formulations is that the chemical stability of the system should result in physical effects rather than chemical reactions. In the previous project we showed that just using four different inputs a wide variety of droplet behaviours emerged, generating a rich compositional space to explore for emergent properties, which was traversed by embodying evolution within the

compositional information of the oil-droplet system. In this project we aimed to take a step further and added a new dimensionality to the described compositional space by embodying evolution directly into the automated device which permitted us to shape the formulation space with different environmental functions.

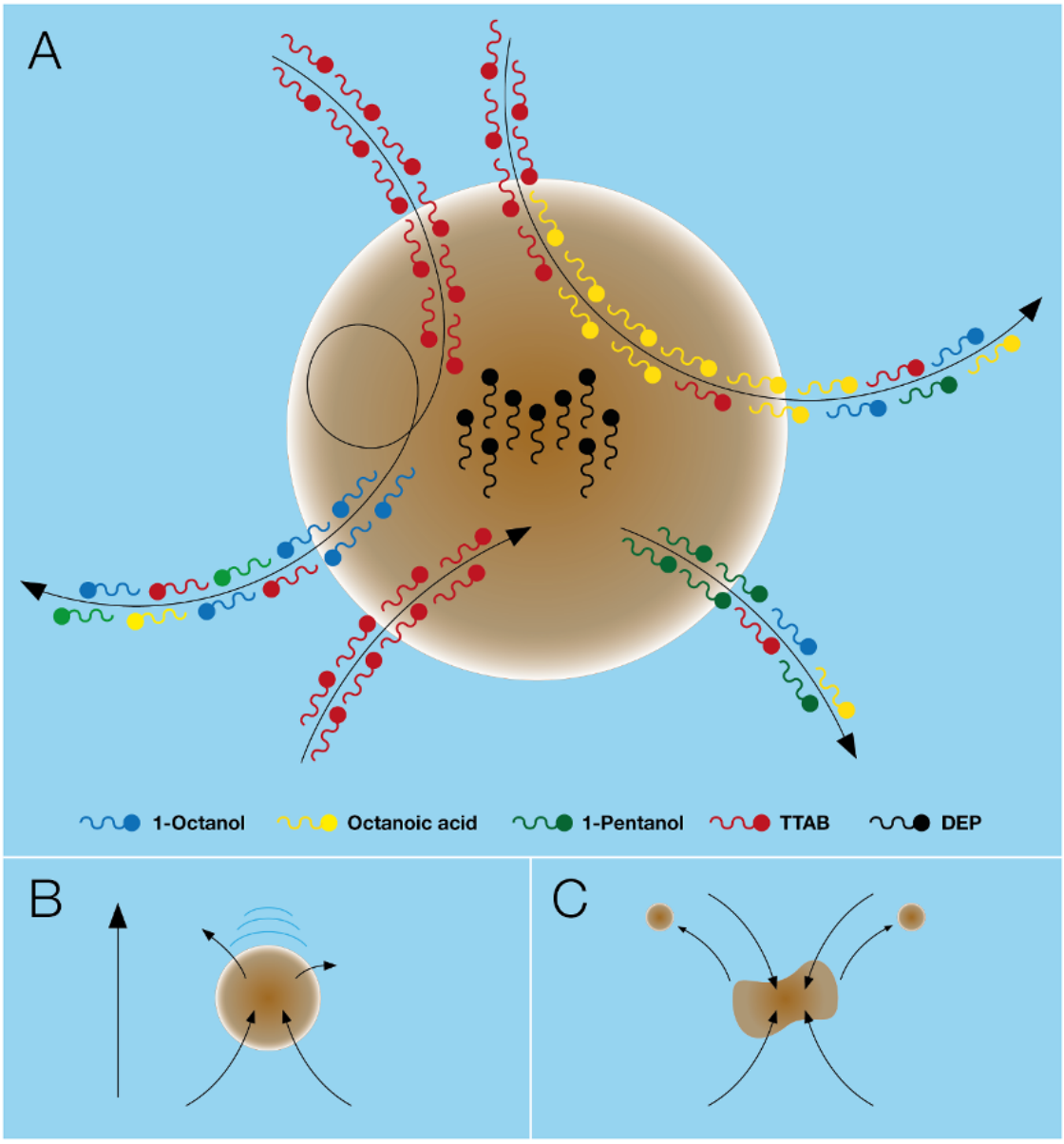


Figure 6.1: Droplets as seen from the surfactant perspective. (A) Our droplet system consisted of oil formulations of 1-octanol, 1-pentanol, octanoic acid, diethyl phthalate (DEP) and tetradecyltrimethylammonium bromide (TTAB) at pH 13 in the aqueous phase. While the oil surfactants were non ionic, TTAB is a cationic surfactant, meaning that TTAB molecules crossed the semi permeable interface until equilibrium. At the same time, the different oils dissolved at different rates into the aqueous phase. Based on the rate that these different gradients were arrived to equilibrium, the droplets would (B) move or (C) divide.

6.3 Platform description

The 3D printed fluidic system was designed using CAD tools and printed using a commercially available 3D Printer[3]. The objective was to encapsulate all the basic functionality from a liquid handling robot into a fully printable monolithic device. The devices were printed using polypropylene (PP) filament, chosen for its high chemical resistance to the compounds and solvents we used. During our research the devices used did not suffer any apparent degradation after hundreds experiments. PP is also one of the cheapest 3D printing filaments available which coupled with the use of 3D printers as fast prototyping tools allowed us to easily modify, iterate, prototype, and manufacture new devices based on different requirements. Therefore, each characteristic of the design could be modified and a new device could be produced promptly with a negligible time or money cost, making our solution suitable for a big range of different experimental and lab requirements.

The main design (Section 5.2.4) contained four inputs for the oil phases which were mixed using a serpentine channel. In order to control the oil mixture, the liquid pumps connected to the device were programmed to dispense the oils at different flow-rates depending on the ratio required by the experiment. The serpentine channel led to an outlet situated at the bottom of the experimental arena which generated droplets into the aqueous phase. In our experiments five 10 µl droplets were generated. Once the droplets were placed into the aqueous phase a video of the resulting droplet behaviour was recorded from above the arena using a camera at a resolution of 800 x 600 pixels at 30 fps. The device's arena also contained two additional inputs for the aqueous phase and cleaning solvent (acetone). In a bottom corner of this area there was a drain point from where the contents from previous experiments were removed. See Figure 6.2.

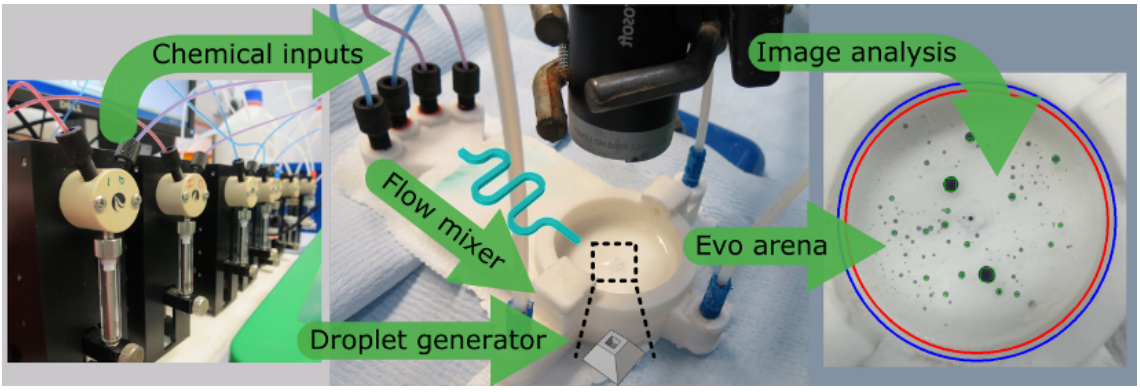


Figure 6.2: Flowbot hardware and software integration. A set of pumps was used to flow a formulation through a serpentine channel in order to mix it. Then, a population of droplets was generated, and a camera recorded its behaviour.

[3]3D Touch from Bits from Bytes

6.4 Evolutionary experiments results

6.4.1 Empty arena

After all the mechanical parts had been individually tested, the device was subjected to an evolutionary experiment in order to test its validity as a liquid handling robot substitute, and its experimental reproducibility. The fitness function used aimed to maximise the number of droplets active at the end of the experiment. A droplet was considered active if it moved and was bigger than a set threshold. Therefore, experiments where the droplets underwent controlled division and movement were given high values.

As explained, a GA was used as the search heuristic. Each GA run executed 10 generations, and each generation had a population of 20 individuals. Each individual was tested five times, and the average was returned as its fitness value. An individual was characterized by its recipe, which was defined as the mixture ratio between the 4 oils. The first generation created its individuals randomly, and successive generations were built using the roulette wheel selection to choose 10 parent candidates, and adding 10 new offspring by stochastically choosing pairs of parents and performing a one-point crossover, followed by a mutation operation. The chosen parents were checked again when testing the new generation, thus focusing on recipes that were reproducible.

The device used during experiments was like the design showed on Figure 5.9 (p.125) or the printed piece shown on Figure 5.12 (p.128). We called it "empty arena" because there was nothing inside, and it was designed to emulate in shape a glass Petri dish like the one used during the Dropbot project (Chapter 3). Our objective when using an arena like this was to replicate the results in said chapter.

A direct comparison between platforms is not possible, because the volumes of the phases are different, and that has a huge impact in the dissolution rate. Also, the way the droplets were generated was dramatically different. The syringe used to place them over the surface of the aqueous phase, while this device generated them inside the aqueous phase pushing them to the top. Finally, the fitness functions are slightly different, because the one we are using now is a kind of combination of what in the previous chapter was defined as "movement" and "division". Therefore, although a direct comparison was not possible, we expected at least to see similar trends.

Results can be seen in Figures 6.3 and 6.4. Figure 6.3 shows a detailed execution where different snapshots of the experiment were taken at generations 1,4,7 and 10. The objective of these snapshots was to visually show how the number of droplets increased through generations. In the case of this execution, the statistical values doubled at the end of the experiment, although a very small growth of around 20% was seen until generation 7.

As we will see during most of the GA execution results, these statistical stagnations around local minima were common in our experiments. We think the cause for them was that we were using small populations of only 20 individuals, paired with only executing 10 generations, and the fact that we picked the 10 best parents without repetitions. These GA conditions constrained the chances of the statisti-

cal values growing steadily through generations, apart from at the start of the run where all the recipes were generated randomly, and some of them were given very low fitness values, and henceforth removed from the pool very fast. Following the actual numerical data we could see that the trajectories usually stagnated until a lucky mutation[4] appeared, and from that point on it improved until arriving at another local minima.

Figure 6.4 shows the rest of the experiments performed for the empty arena environment. All of them with the GA conditions as described before. In all the cases when comparing the first and last generation a growth can be seen, usually around doubling the initial average numbers, but none of them were clean growing evolutionary trajectories. In any case, this kind of "dirty" growing linear plots were also seen in the previous chapter, and we think that when working with real world experimental data the results are never perfect, and these type of variations can be expected.

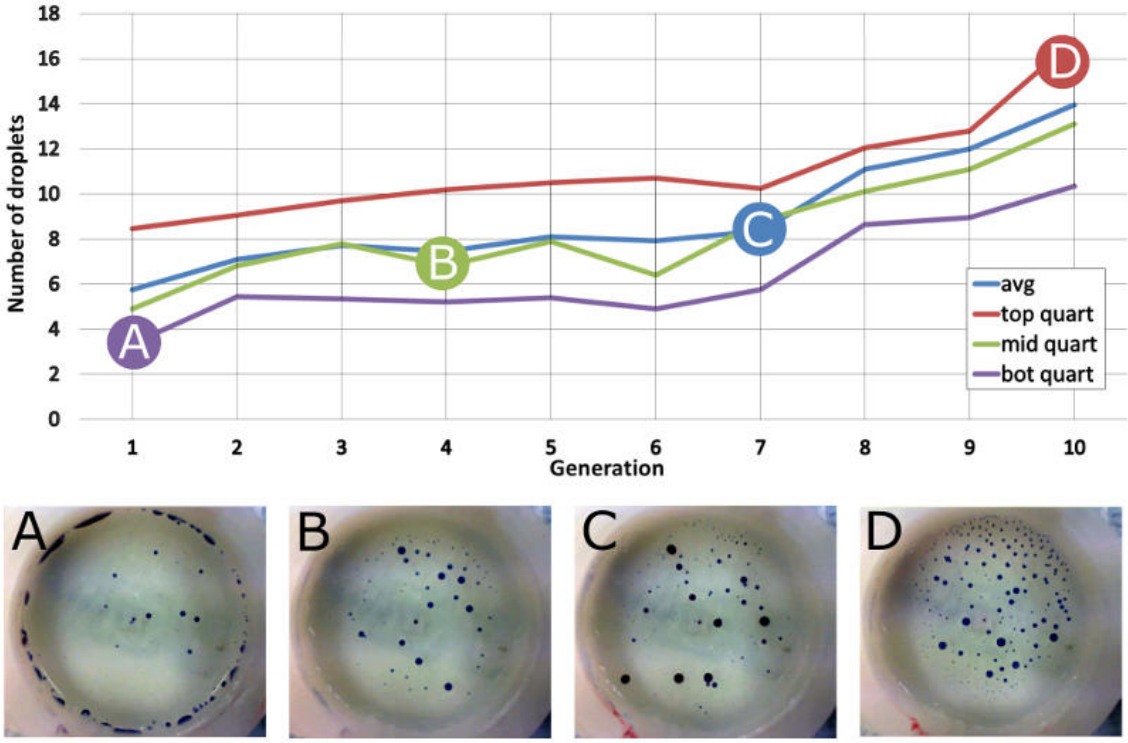


Figure 6.3: Fitness increase over successive generations and platform validation. **Top:** Linear plot showing the change in fitness (evolutionary trajectory) over each successive generation of experiments for the defined fitness function (droplet activity). **Bottom:** Screenshots taken using the last frame of the video experiments. Through each picture it can be visually seen that the number of droplets increases.

Figure 6.5 shows the weighted average genome evolution for the GA run third and fourth using the empty environment. In both cases 1-pentanol was the strongest

[4]Because we were using only 20 individuals per generation, a "lucky mutation" adding a new individual into the pool with a fitness of 15, for example, would mean that the global average would raise by 0.75, which as can be seen in the plots, can be a significant raise.

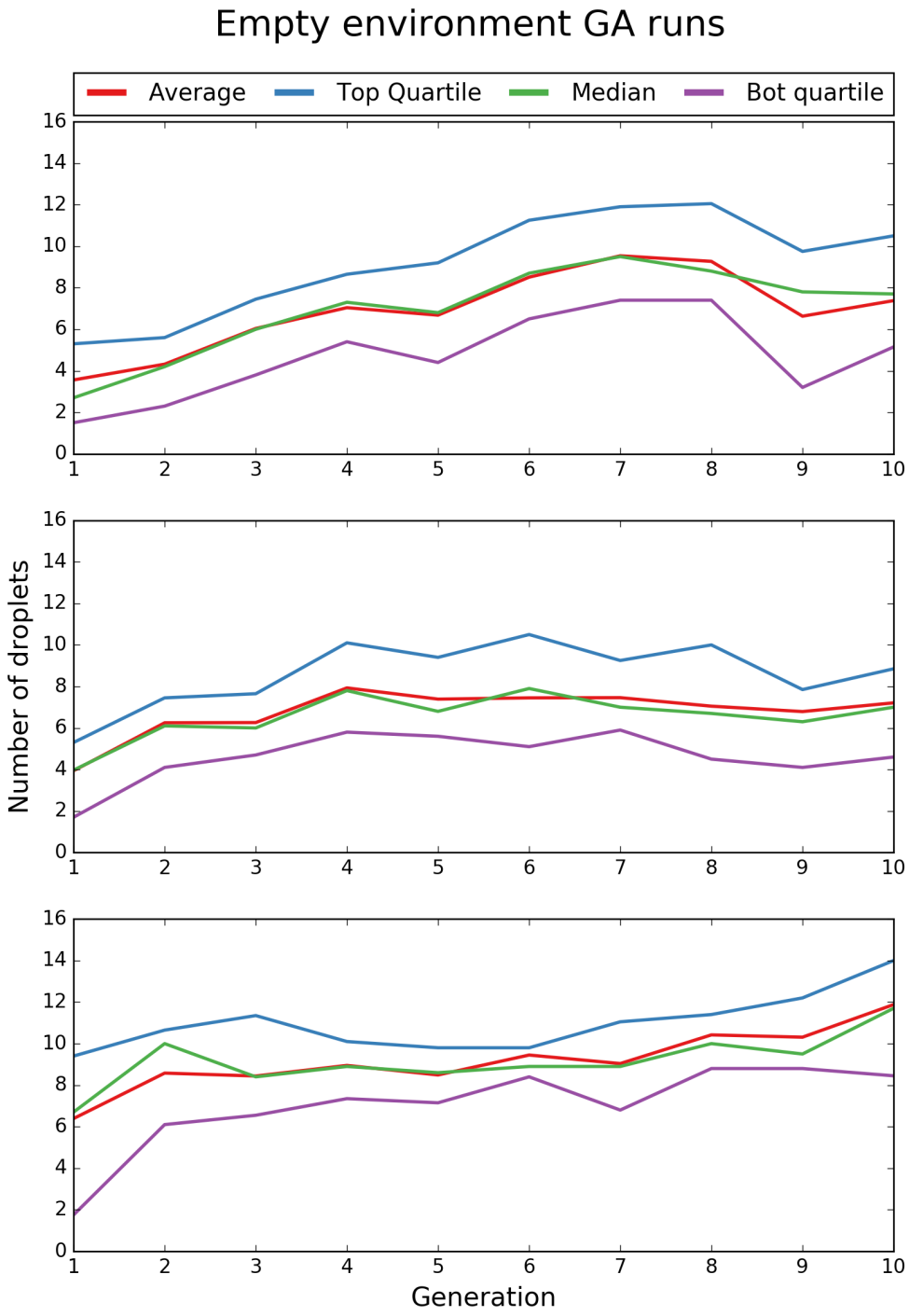


Figure 6.4: Flowbot empty arena genetic algorithm runs 1,2 and 4. The third execution is detailed on Figure 6.3. Each of the generations represents 20 individuals.

component at generation 10, and octanoic acid the lowest. One of the executions kept 1-octanol active, while the other also removed it.

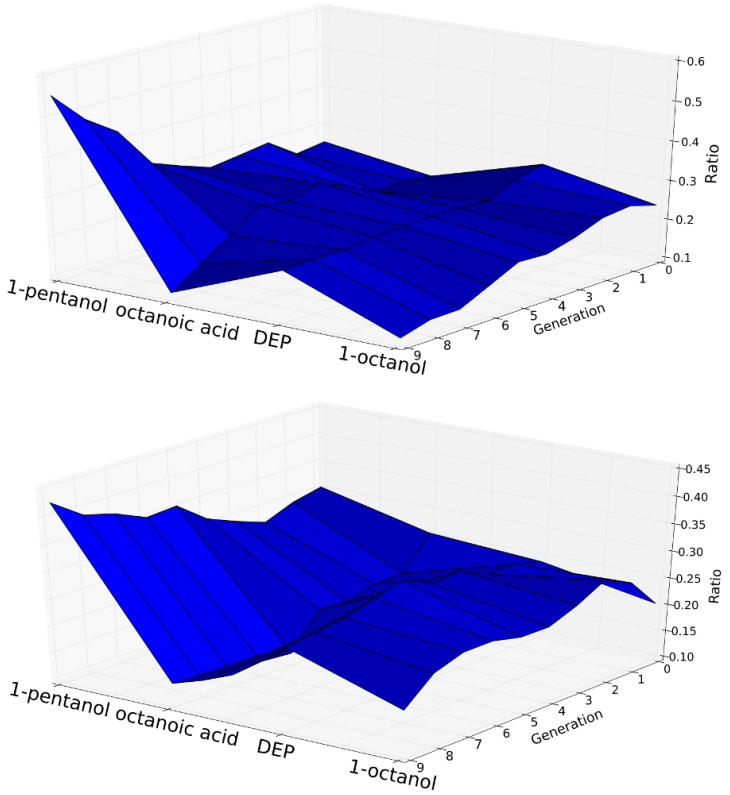


Figure 6.5: Weighted average genome evolution for the third (Figure 6.3) and fourth (Figure 6.4 bottom plot) GA run with an empty environment. The ratio in the Y axis is the average ratio per component.

In all the GA runs it can unequivocally be seen that populations of droplets were evolved through generations in order to maximise the defined behaviour, because the number of droplets nearly doubled at the end of the experiment. These results were very similar to the ones showed in previous project (Section 4.4.3). In particular the genomic evolution plots are very similar, because in both platforms 1-pentanol was maximized, and then the rest was shared between 1-octanol or DEP, while octanoic acid was removed.

These results demonstrated that our 3D printed device not only could replicate the capabilities of a liquid handling robot, but also perform evolutionary experiments.

6.4.2 Pillars arena

Taking advantage of the novel modular design of our device, the open arena could be transformed into different environments to perform evolutionary experiments. This way, not only the oil-droplet compositional information impacted the behavioural phenotype, but also the characteristics of the environment the droplets resided in, because by modifying the environmental parameters through different experiments

we added a new degree of freedom to our evolutionary experiments, similar to how nature shapes the individuals that inhabit it.

In the first custom environment the arena was densely filled with pillars, see Figure 5.17 (p.140). The pillars had a circular shape and a diameter of 2 mm, which is roughly the size of half droplet[5]. Its height was also high enough to be above water level. Our initial hypothesis for designing such environment was that the droplets would move around, bump into the pillars and divide. Therefore, initially before doing any experiment we thought that the pillars would have a positive effect on our fitness function in terms of producing higher fitness values.

Once we started doing experiments in this new arena we realised that the results were different to what we expected. Some of the formulations did move, bump into the pillars and divide, but some of them would actually be attracted to the pillars, where the droplets would attach and remain inactive. Another effect of the pillars we did not consider initially is that if we consider the water surface profile, when using an empty arena we can consider that it is almost flat, generating a meniscus that in areas of 4 cm diameter is almost negligible. On the other hand, in our new arena each of the pillars was pulling the water surface, generating a non uniform profile with small valleys in the areas between the pillars. This had the effect of the droplets accumulating in these areas, especially near the of the experiment when they were running out of energy, thus inhibiting their movement. Hence, while initially we thought that the pillars would produce higher fitness values, it was actually the opposite, because they would directly "kill" the droplets by trapping them, or inhibit their movement. Some of these behaviours can be seen on Section 6.5.

The GA conditions were the same as in the previous experiment, and two separate GA runs were executed, see Figures 6.6 and 6.7. In particular, Figure 6.6 shows different snapshots of the evolutionary process where different formulations can be seen as they are optimised through the GA. It can be seen during the first screen how the droplets were trapped into the pillars, and in some of the other screens there can be seen how the droplets accumulated in the free space between pillars.

Nevertheless, in both runs the fitness value improved through generations doubling its values at the end of the experiment, proving that our device was able to embody evolution into it by shaping the evolutionary trajectory in different ways. We could also see, as explained before, that the pillars had an actual effect on the droplets behaviours.

Finally, Figure 6.8 shows the fitness weighted average genome evolution through generations. This surface map looks very similar to the one showed before (Figure 6.5) in terms of the final genome. In both cases it was dominated by 1-Pentanol although now DEP had a stronger presence. It can also be seen that in this case the genome evolved much faster, and that right at the start the fitness of some experiments was so bad that the evolution was already biased towards a small set of the initial formulations.

[5]The size of a droplet depends on its composition, but roughly on average they had a diameter of around 4 to 5 mm

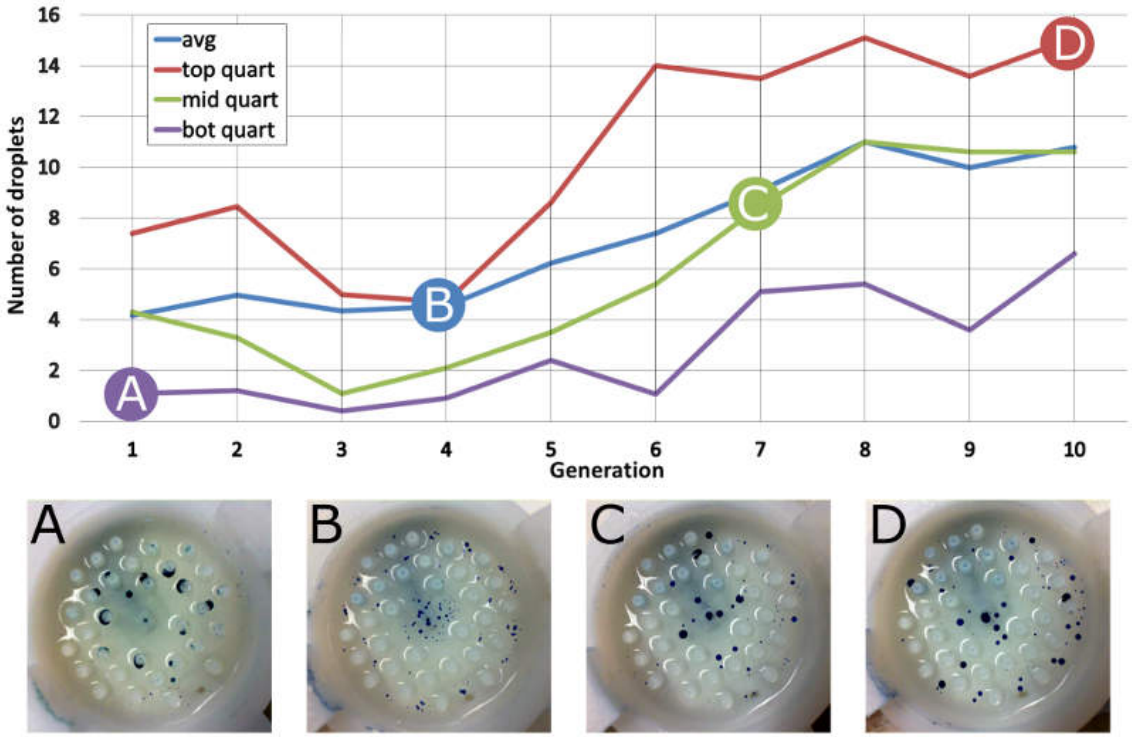


Figure 6.6: Fitness increase over successive generations when using the pillars arena. **Top:** Linear plot showing the change in fitness (evolutionary trajectory) over each successive generation of experiments for the defined fitness function (droplet activity). **Bottom:** Using the pillars environment, successive pictures were taken to represent the increasing number of droplets through generations. A second execution was performed, and the results can be seen in Figure 6.7.

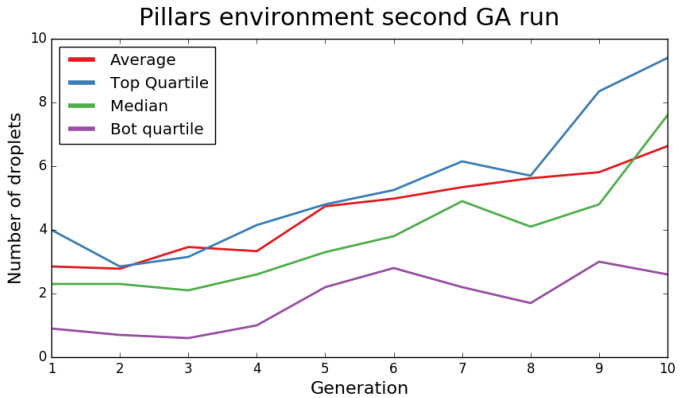


Figure 6.7: Flowbot pillars arena genetic algorithm second run. The first execution was detailed in Figure 6.6. Each of the generations represents 20 individuals.

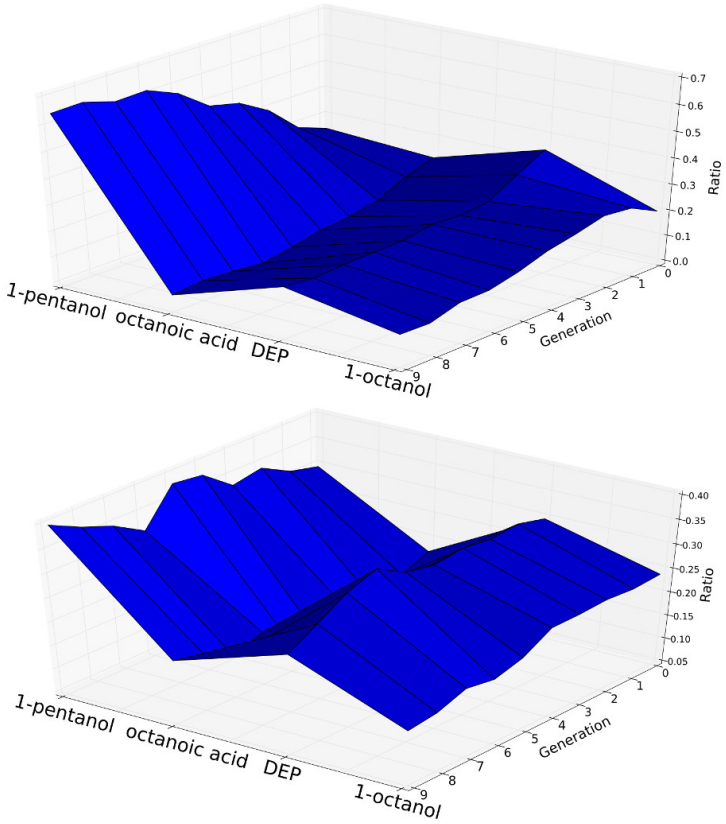


Figure 6.8: Weighted average genome evolution for the first (Figure 6.6) and second (Figure 6.7) GA run with the pillars environment. The ratio in the Y axis is the average ratio per component.

6.4.3 Caves arena

After showing that our system was capable of undergoing evolution in an environment radically different to a Petri dish, we decided to test it again with another environment, but this time the environment was generated by an algorithm.

Section 5.5 in the previous chapter (p.140) already explained the design process, and the device we used was exactly like the one shown in Figure 5.20 (p.143). Our initial hypothesis was that the cave-like formations would trap the droplets inside them, inhibiting their movement and division. Thus, we thought this environment would have a negative impact.

Once we performed the experiments the behaviours we expected did not happen, maybe because our caves were not deep enough for the droplets to go in[6]. The effect of the caves in the droplets was similar to the ones we had with pillars. On one hand, the droplets would attach to the plastic obstacles, and their attachment now was stronger because there was more continuous surface. On the other hand, the caves would change the surface of the water creating valleys like before. The main difference is that the valleys were now bigger because there was more free area between obstacles.

Two different GA runs were executed with the same conditions as before. Results can be seen on Figure 6.9. As before, both GA runs were successful at optimising the recipe for the given fitness function, and in both cases the initial statistical values were doubled at least. Although we expected a successful result based on the previous experiments, it was very interesting to know that we could create new environments, and that a GA would be able to optimise a formulation for them.

Finally, about the fitness weighted genome evolution surfaces (Figure 6.10), one of the GA executions was similar to the ones showed before in that right from the first generation the genomes were clearly biased towards a given recipe. The other one was more interesting, and was one of the few executions in the global of our experiments were there was a strong presence of 1-octanol. It is also interesting to note that the first GA run, which is the one with the higher values, was also the one biased from the start. While looking the other run, the surface seemed to be around a local minima for a while, outputting low values, and in general its statistical values are lower than the previous one.

6.4.4 Hybrid GA runs with environmental changes

In order to emulate how rapid environmental changes affect evolution, a hybrid GA experimental run was performed where the initial 10 generations used the arena with an empty environment, the following 10 generations used the arena modified to contain the pillars environment as described, and the last 10 generations used the arena designed using a L-system, see Figure 6.11. It can be seen that after the first change the evolutionary trajectories dropped nearly by half their value after the environment was changed, proving that our environmental changes had an impact over the droplet's behaviour, while the second change of arena had a slight positive effect on the evolutionary trajectories.

[6] if we were to repeat these experiments, we would design deeper caves.

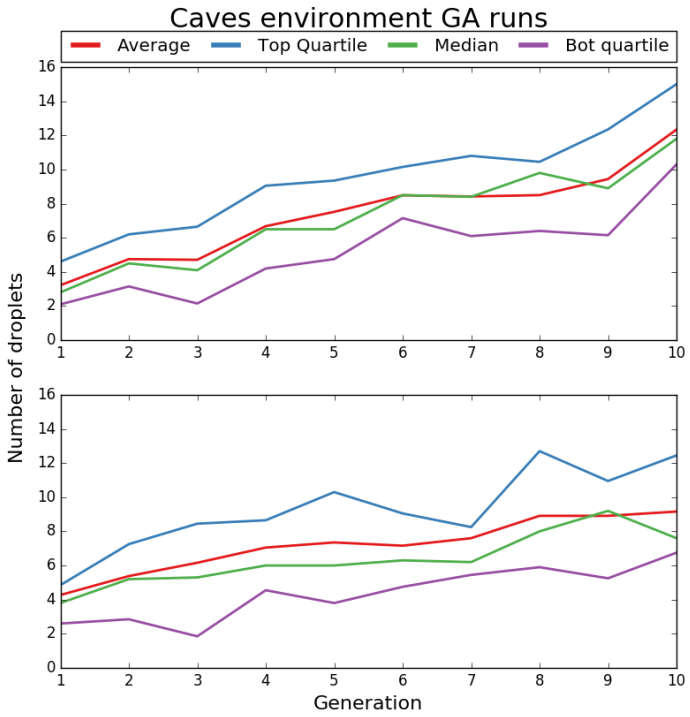


Figure 6.9: Flowbot caves arena genetic algorithm runs. Each of the generations represents 20 individuals.

Specifically, the data model created as explained in the experimental chapter, Section 8.2.11 (p.197) illustrates that the fitness landscape of the environments using pillars or generated with the L-system were a subset of the environment without obstacles. These results determined that the evolved genotypes under the environment using obstacles were a subset of the genotypes under the empty arena, showing that our rapid modifications acted like a filter over the population of individuals.

Figure 6.12 shows the fitness weighted average genome through generations in this hybrid GA run. There it can be seen how the genome changed significantly after the environment was swapped the first time, and this might relate to the drop of statistical values we saw before. In the next change of environment there was not a big genome variation, although the quantity of 1-octanol increased, but it is impossible to say if this increase was caused by the new environment or if it was following a trend coming from the previous environment. For the same reason, it is impossible to say if the increase in fitness values that we can see in Figure 6.11 during generation 20 and 21 was caused by the new environment, or if it was just following the previous trend.

Figure 6.13 shows another hybrid GA run, in this case only focusing in the transition between empty to pillars. The values dropped as expected, although in this case it took them much longer to recover. This figure focuses in particular in the behaviour of some of the recipes before and after the pillars were added. The pictures shown relate the 10 formulations from generation 11 which were taken from generation 10 as the best parents. There it can be seen that in most of the cases the droplets got trapped into the pillars, and that explains why they dropped in fitness

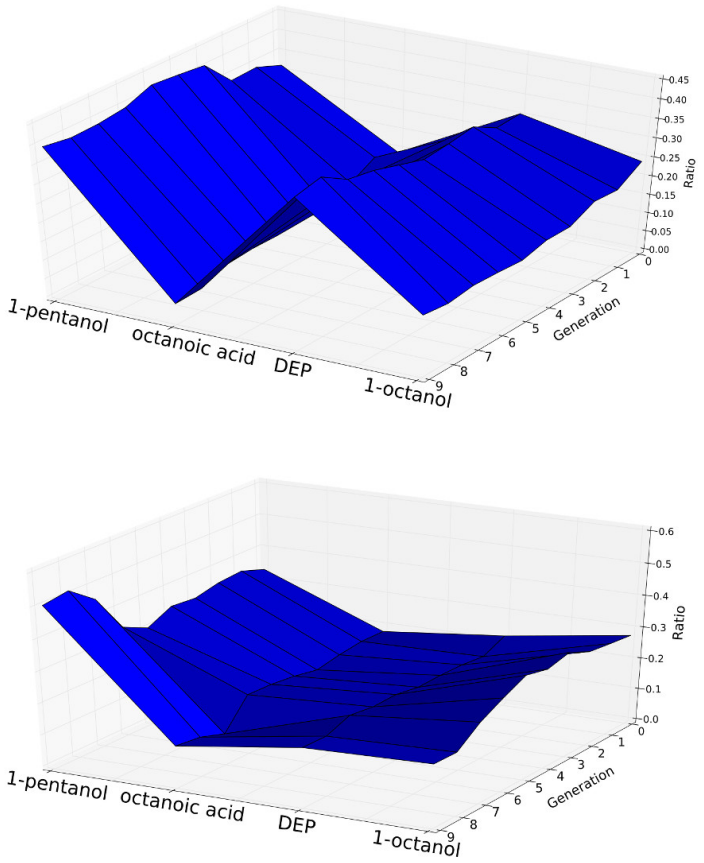


Figure 6.10: Weighted average genome evolution for the two GA runs (Figure 6.9) with the caves environment. The ratio in the Y axis is the average ratio per component.

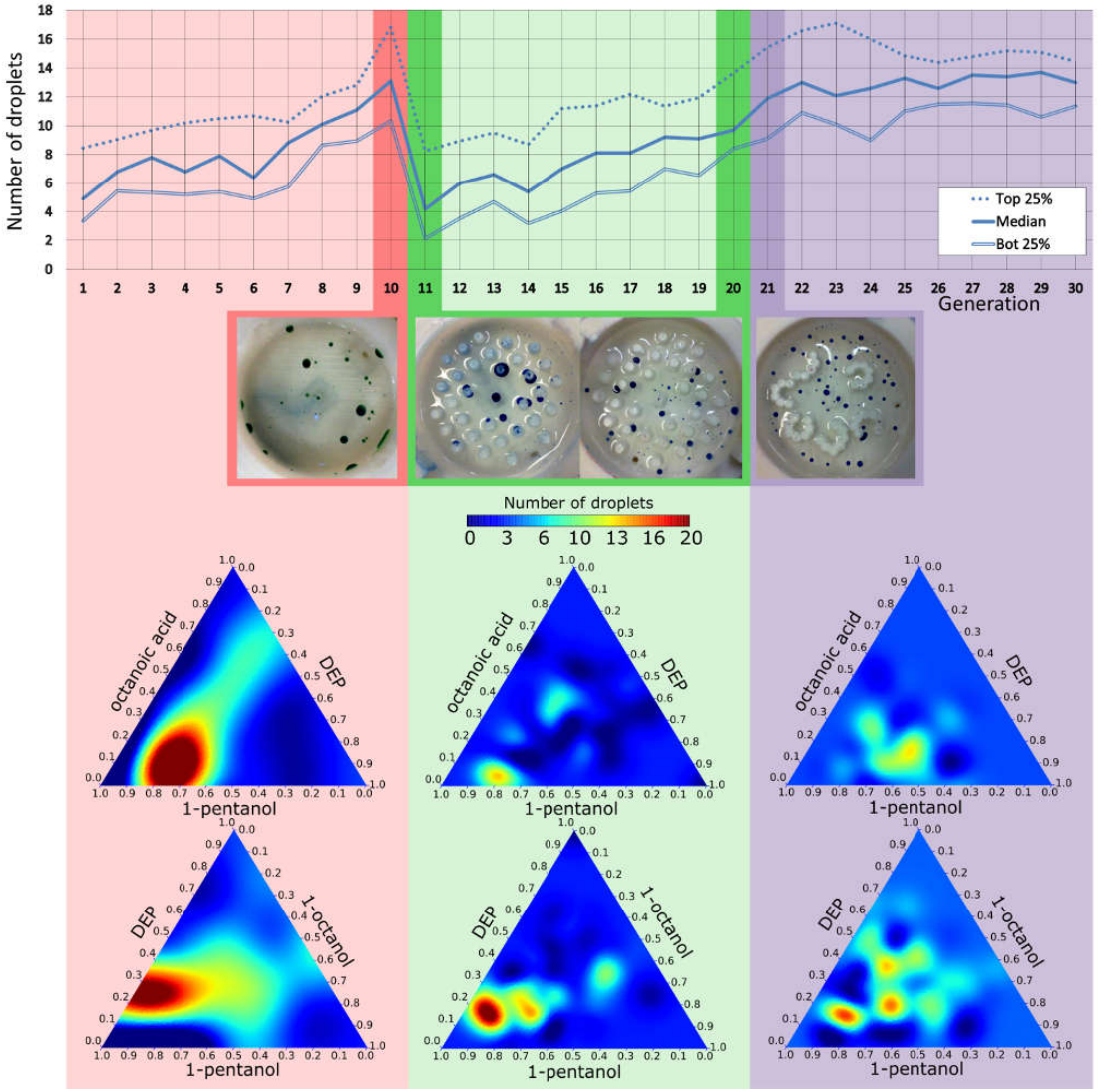


Figure 6.11: Flowbot hybrid GA run enviromental change data modelling. **Top:** Linear chart plotting the statistical values of each generation through time. The first ten generations used an empty arena, the following ten generations used an arena filled with pillars, and the last 10 generations used the arena generated using a L-system. The arena was swapped between the 10th and 11th generation, where a drop in the statistical values can be seen, and between generations 20 and 21. **Bottom:** Sections of the ternary fitness maps derived from the experimental data.

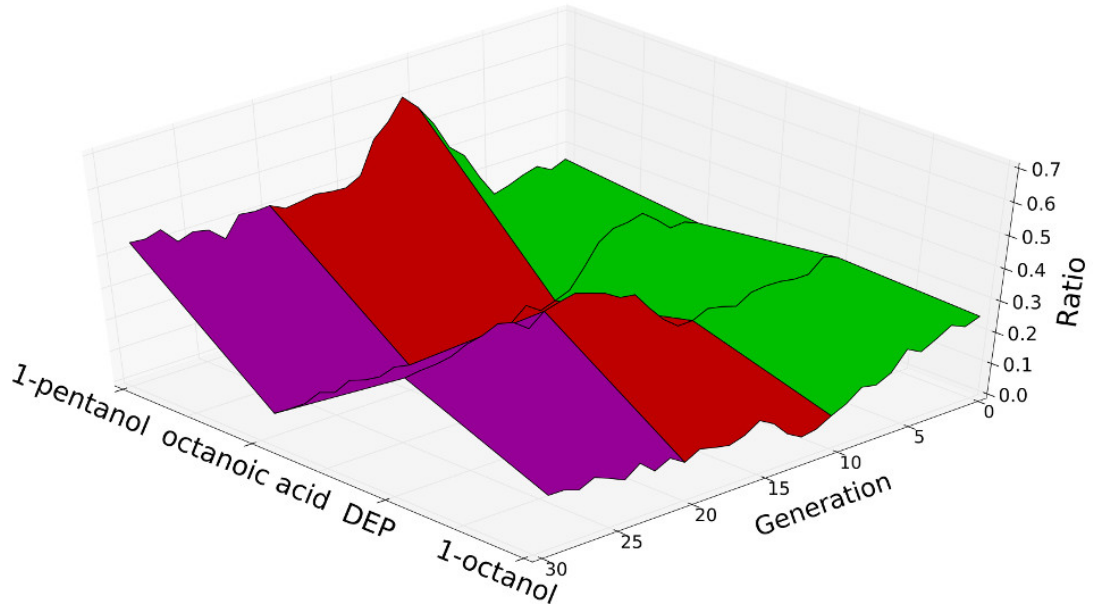


Figure 6.12: Weighted average genome evolution for the hybrid GA run using the three environments in a consecutive way. The green area marks when the empty arena was used. The red area marks where the pillars arena was used. The purple area marks when the procedurally generated arena was used.

value.

	Octanol	DEP	Octanoic	Pentanol	F. Empty	F. Pillars
A	.35	.17	.14	.34	8.8	.4
B	.41	.18	.0	.4	13.6	1.4
C	.41	.16	.02	.41	13.2	.6
D	.31	.18	.04	.46	11.8	.0
E	.28	.25	.06	.41	10.2	6.8
F	.25	.28	.07	.4	11.4	10.8
G	.29	.19	.0	.52	15	3.6
H	.37	.17	.22	.23	5.8	.6
I	.44	.14	.0	.42	8.8	.0
J	.27	.24	.14	.35	7.8	7.8

Table 6.1: [F. = Fitness value] The data represented in this table is directly related to Figure 6.13. The letters in the left column relate to the letters used in that figure. The last two columns describe the fitness values for both the empty environment in generation 10, and pillars environment in generation 11. The green highlight shows the formulations where the fitness value did not change. In orange it is highlighted when there was a drop, but the value was still decent. All the other ones left blank are the formulations where the fitness value after changing the environment dropped significantly.

.

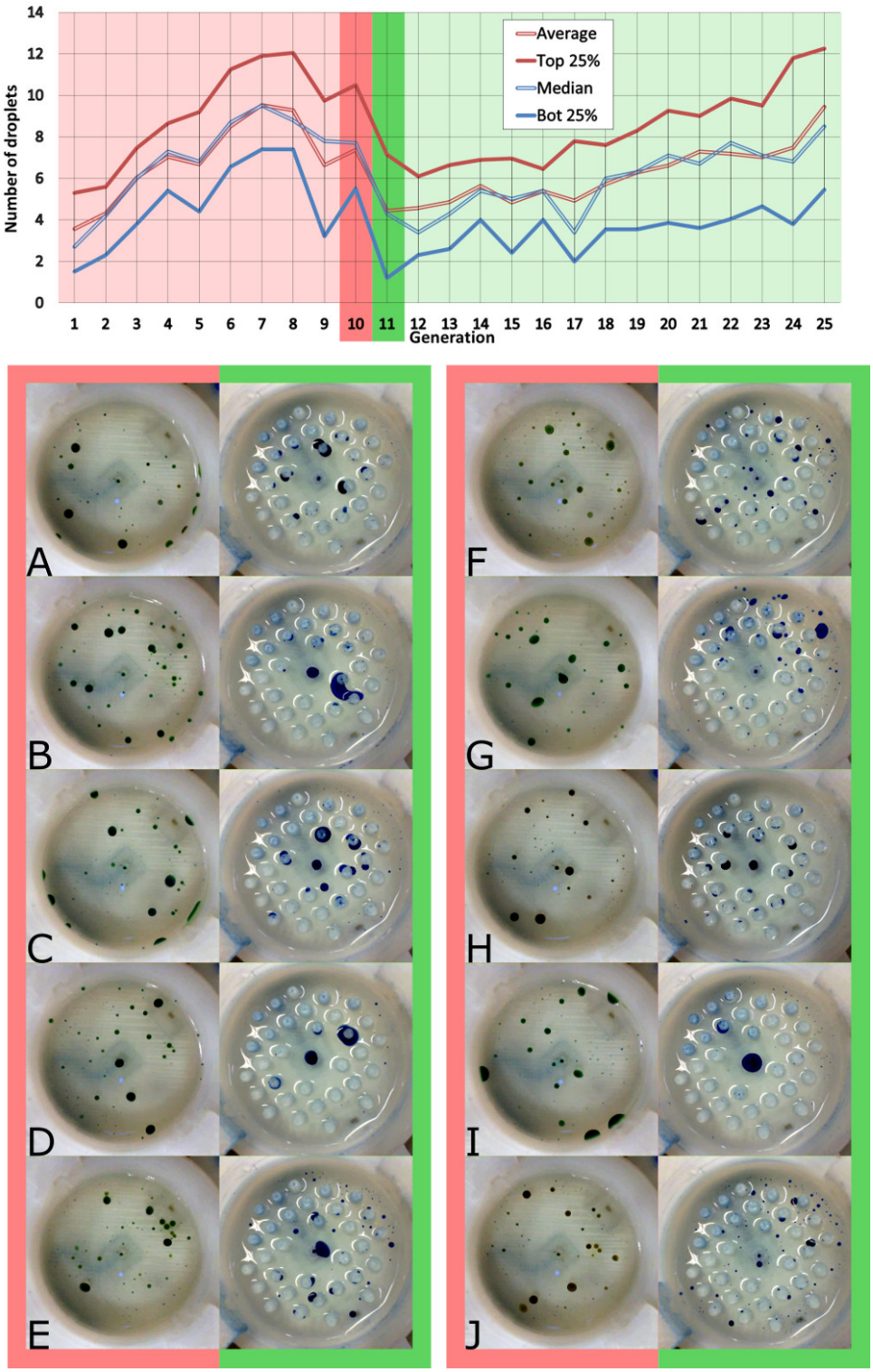


Figure 6.13: Hybrid GA run focusing on the behaviour change between generations 10 and 11. In this figure we can see the ten chosen parents to build generation 11 from generation 10, compared one against another. It can be seen that most of the best recipes during generation 10 get attached and attach to the pillars in generation 11. This figure is complemented with Table 6.1.

6.4.5 Experimental reproducibility and control tests

Figures 6.11 and 6.12 were the key result to this project, where we showed that by changing the environment we could change the fitness function. There are, though, two big questions about the experimental reproducibility:

- What if these results were just luck and it did not happen again?
- What if any new environment would produce different values because all the devices are slightly different? Thus the environment itself does not matter, it is the whole device what makes a difference.

In order to answer these questions we tried to check them in series of follow-up experiments. The first experiment was performed using the same conditions as with Figure 6.11. The main difference is that in this case we only explored the first transition. Figure 6.14 shows the result, and again, once the environment was swapped we could see a drop in the statistical values. Therefore, when swapping from empty arena to "pillars" we saw three times that the evolutionary trajectories dropped dramatically. Although it might still be luck, we considered that three repetitions are enough to prove our point.

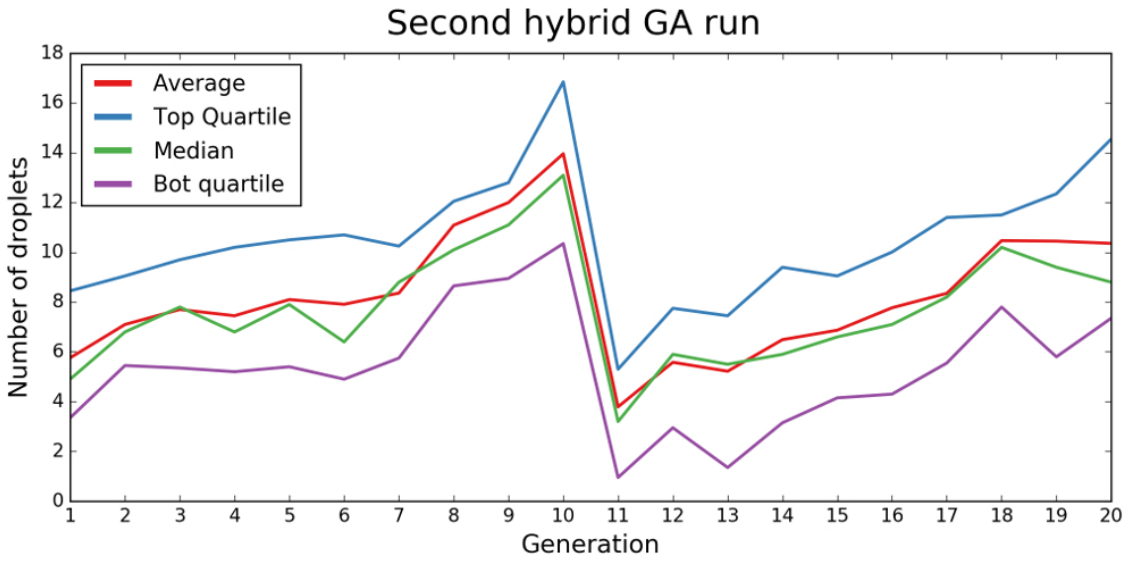


Figure 6.14: Second hybrid GA run. This linear plot represents the change in the evolutionary trajectories caused by changing the environment between generations. As before, the change was performed during generations and 10 and 11, and as it can be seen, the fitness value dropped by more than half. All the other GA conditions were exactly the same as in previous experiments.

In order to answer the second question a series of short experiments were performed, where data from the previous experiments was used, and a few new generations were produced to continue studying the impact of the environmental changes upon the statistical evolutionary trajectories, see Figure 6.15.

The two first experiments, top row, aimed to check if the changes we were seeing were caused by the obstacles or just by swapping by a new device. In both of those

experiments, during the first 10 generations an empty device was used, and then two new generations were executed using a different device, also with an empty environment. It can be seen that there is almost no difference, and that in both cases it kept a very similar value. It is also important to note that in one case the initial 10 generations were done using a device printed with translucent PP, and the extra two generations using a device with white PP, while in the other test it was the other way around. Thus, we also check that the dye of the plastic did not have an effect.

In the next experiment, middle row, we did the opposite experiment as in the previous section. Here, a pillar environment was swapped by an empty one, while before it was the other way around. In this case, the statistical values grew instead of going down. So this reaffirmed our hypothesis about the environment having an impact on evolution.

Finally, in the last row there can be seen the last series of experiments, very similar to the ones executed in the previous row, although in this case instead of going from empty to pillars it went from empty to caves. Interestingly, the statistical values also dropped, but not as much as before. This is interesting when considering what happened on Figure 6.11, where between generations 20 and 21 a pillars environment was swapped by one with caves, and the statistical values went up. These new results seem to explain that the caves environment had a less negative effect than the pillars one, and that is why in that case it went up, and in this case it does not drop by that much. We did not develop more studies about this, so this is just a possible hypothesis.

6.4.6 Additional statistical data

After 10 generations of individual GA runs, the fitness weighted average genome (WG) for each of the environments and their standard deviation were (octanol%, dep%, octanoic%, pentanol%): (15±11, 23±7, 13±11, 49±14) for the empty environment, (13±7, 22±5, 10±11, 55±15) for the pillars environment, and (17±9, 30±12, 8±10, 44±11) for the procedurally generated environment. A one-way ANOVA test comparing the WG of each of the arenas returns statistical significant p-values for the DEP (0.01) and 1-pentanol (0.002) components. Their last generation fitness was also significantly different (0.004).

The WG for the pillars environment after 20 generations using the empty environment during the first 10 generations was (9±6, 37±6, 8±8, 47±9). An ANOVA test comparing this population with the one when only using the pillars environment through all the GA run shows significant p-values in all the components (0.0001, 1.5e-14, 0.03, 0.001).

Finally, the WG when the first 10 generations used the empty environment, the following 10 the pillars environment, and the last 10 used the procedurally generated environment (Fig. 5) was (16±11, 36±6, 6±5, 43±10). An ANOVA test comparing the last population of this GA run against the one obtained as a single environment execution shows significant p-values in the DEP (0.03) and 1-pentanol components (0.008).

It is interesting how in every case the final genome differed when environmental

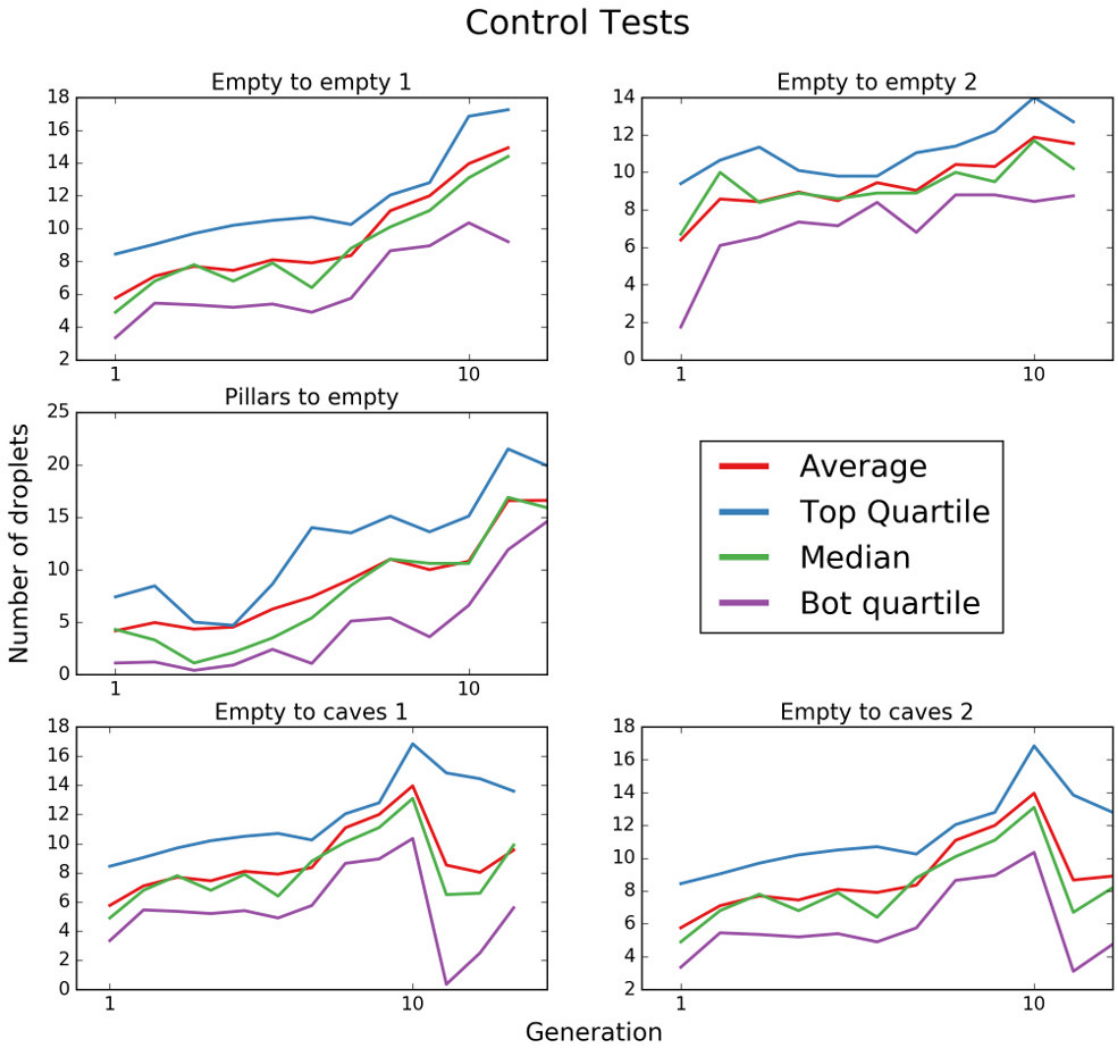


Figure 6.15: Flowbot experimental control tests. Five short experiments were performed reusing data from the previous experiments and testing how the environment shaped the statistical evolutionary trajectories.

changes were introduced, as opposed to the experiments when a single device was used. This suggest that the evolutionary pathway was influenced by the environmental history, effectively driving the evolutionary process into a different fitness niche than the immutable-environment. It confirms that the environment and its evolution through time plays an active role in the development and expression of our droplet system. Thus, considering the environment itself as a variable might be a promising approach for the optimization of complex systems.

6.5 Lattice search

As with the previous project, we also performed a lattice search as explained in the experimental chapter (Section 5.4.2.2, p.135). While in the previous project the lattice search was performed in order to test the robustness of the platform, in this case it had the objective of fully exploring the formulation space, especially its corners and the portions never tested by the genetic algorithm. As explained, the four different oils were taken into combinations of pairs, threes and fours, with a granularity of a 10%, exploring in total 282 different oil formulations. Each formulation was tested 5 times, generating 1410 videos.

This lattice search was carried for each of the different arenas. Thus, not only was it useful to traverse unexplored portions of the space, but also to compare the exact same formulation being tested into different arenas. Here we will detail the results obtained for the "pillars" environment, because the lattice search performed for the "empty" environment was very similar to the one done in the previous project (p.89, Section 4.4.1), and the one done for the "caves" environment was very similar to the one that will be described now.

Focusing on the pillars environment, it confirmed some of the behaviours seen during the evolutionary experiments: (a) the pillars would act as droplet divider when moving droplets bumped into them, although in some other cases (b) the pillars would attract the droplets and inhibit its moving behaviour, although (c) some droplet formulations would unhook themselves after a set time and then continue moving, also (d) the pillars pulled the water surface creating valleys between them where populations of low energy droplets locate, and (e) in some cases the pillars would trap the droplets in a way that they lost their integrity. See Figure 6.16.

6.6 Relation between droplets and obstacles

During this research we did not perform any study as to why the droplets behaved the way they did in the presence of obstacles. The material used was PP, which should be inert, so the answers we can offer are just about physical phenomena where the droplets just move around and bump into the obstacles. It was surprising than more often than not the droplets would get attached to the pillars instead, and our explanation for this would be very similar as to why they would splash against the Petri dish walls in the previous chapter. We think they were just doing so to lower their inter-facial tension.

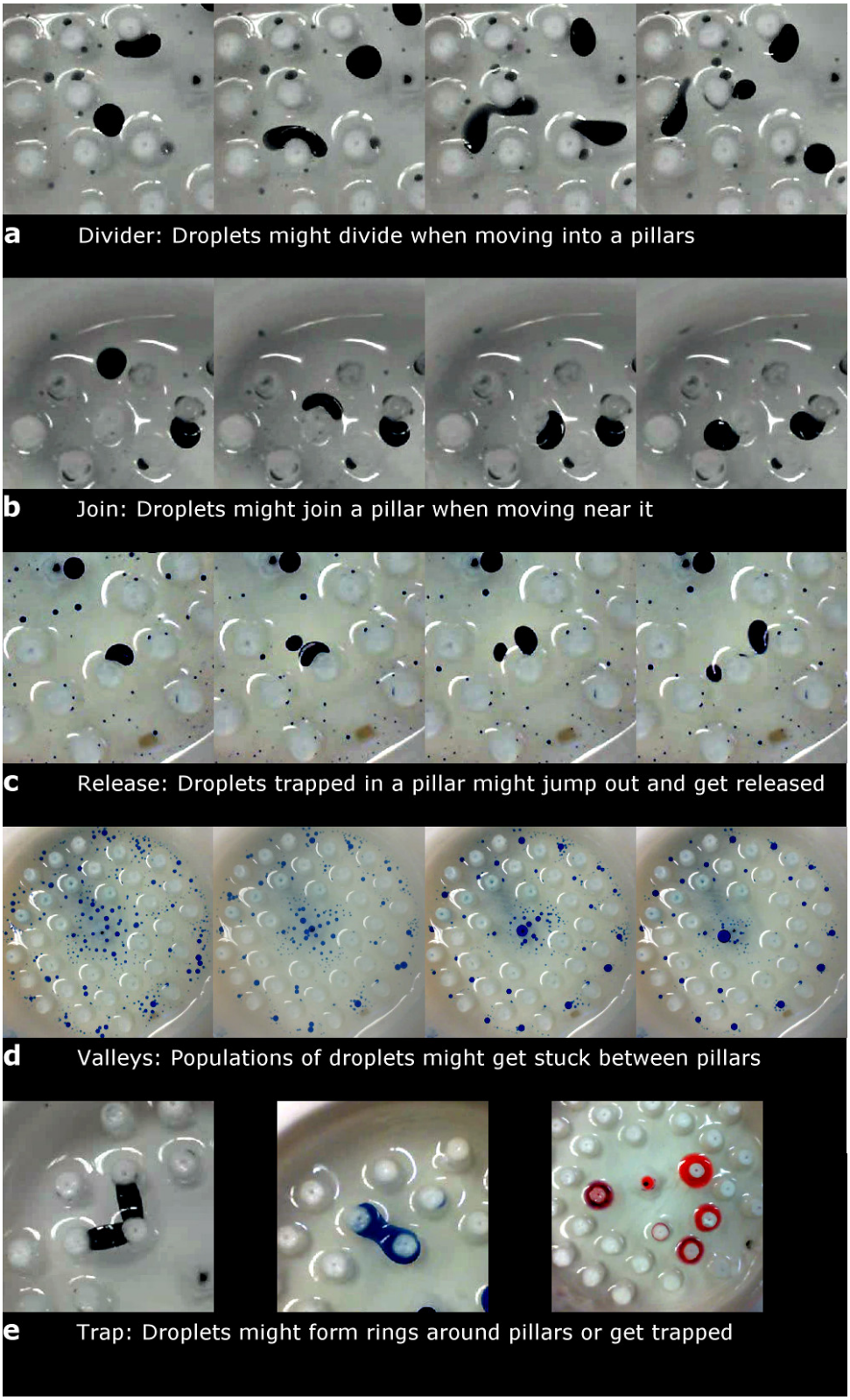


Figure 6.16: Droplet interactions with the obstacles observed during the lattice search. Photographs of the droplet behaviours as a function of time (left to right) except for the (e) where it represents a single photograph.

It would be interesting to continue this study and answer exactly why did it happen, and to analyse if there was any chemistry mechanism undergoing. Something very interesting we saw is that very rarely different droplets would go to the same pillar[7], but instead each droplet would go and attach to a different pillar.

6.7 Discussion and future work

By using a 3D printed fluidic device with programmable environments we were able to embody evolution into an experimental monolithic set-up, demonstrating that the system evolved as a whole. This means that it was the interplay between chemistry and the generated physical obstacles what allowed us to explore the potential of oil droplets to evolve under a set of given conditions, and then adapt and continue evolving once these conditions were modified. This way, we have shown that simple oil in water droplet formulations are a viable model for evolvable protocells, because even though they do not contain any kind of sequential information, using our fluidic device we managed to assist them to evolve following a defined fitness function. Moreover, we showed that populations of droplets can overcome rapid environmental changes and continue their evolutionary process in the new scenario, akin to how living entities adapt to new habitats. Not only our platform can be used to optimise oil in water droplet formulations, but it can be easily extended incrementing the number of inputs to use different reagents or changing the experimental arena to perform different experiments. Furthermore, because the device is completely fluidic, its functionality can be expanded adding elements from flow chemistry or microfluidics, especially integrating new flow analytical techniques to increase the number of system outputs. As a result, we hope this work can open a new line of study about how coupling minimal information inputs with chemistry and algorithms can lead to spontaneously emerging complex chemical systems.

In summary, we believe that embodying evolution into an automated platform as presented could result in a viable new perspective in the study of the intersection between abiogenesis, robotics and the design of new artificial life forms, in particular the ones with a plausible extrapolation to the origin of life. The system as described could be used to research possible pathways from a given chemical soup to an autonomously evolving system, even more, by embodying evolutionary processes into it we believe that an unlimited number of pathways can be generated, which would result in a system capable of open-ended evolution. Also, although in the current design the fitness function was predefined, another very important objective will be to explore the "novelty", that is, the emergence of unpredicted behaviours. Finally, although the results as described show that most of the characteristics of "life" are still dependent on the automated device, we have taken a step forward into removing stepwise these dependencies, a process we aim to continue in the following evolutionary experiments until achieving a fully autonomous chemical evolvable system. Therefore, our objective is to continue removing technology by embodying it into the experimental system while keeping the consistency of the initial hypothesis, focusing on the creation of artificial life entities based on abiogenesis, driven by a thesis that

[7]We never saw this happening

is chemically agnostic.

Although a lot of data has been presented, and around 20,000 individual experiments were executed, we still feel we would need more data to make our point stronger. In particular, if we have had more time we would have liked to develop more hybrid GA runs where different environments are used at different stages of the experiment. Something very interesting would be, for example, to repeat the triple hybrid GA run but the other way around, starting from caves, then pillars, then empty, and then compare the final results and formulations with the ones presented here.

There is also a lot of data missing about exactly how much the different environments shaped evolution, in particular what was exactly their role. Our answer here was limited by the fact that we could only test three different environments, so it would be interesting to generate new environments and test them. Also, as said we ignored the chemical factor of the obstacles, and we only considered them as inert entities, but what if the polymer (PP) also played some role with the different oils or surfactants? What if the obstacles became contaminated through the experiments? It would be very interesting to test these possibilities.

There was also a lot of work planned about doing bigger modifications to the device that we did not have the time to execute, for example adding new oil inputs. As before, we would like to know what would happen if we perform a hybrid GA run where the first 10 generations used four inputs, and then the next one use the same environment but with five inputs. Would the new input be assimilated into the system? Would it help the fitness function? In the same way, we saw that when going from empty to pillars the statistical values dropped. We would have liked to test that again, but adding a new input, and checking if the new input helps to soften this drop.

Another related experiment, but more from a theoretical point of view, would be to study the role of neutral DNA on evolution. It is know that most of our DNA is inactive, but it is also believed that it plays a big factor in evolution. We would be interested on performing a simple experiments, where the number of inputs is four as before, but where the genome size is bigger, and check if the new neutral genes help in any way, in particular when the environments were swapped.

As for the results in general, the evolutionary trajectories were not as clean as expected. This could be because the GA conditions were very limited (only 10 generations and only 20 individuals per population), but also it might be because the degree of contamination or noise was bigger than expected. This is why a deep study of the platform would be required in order to quantify if there is any contamination going on, or how much noise it is produced. We must say that in our experience while doing the experiments most of the variability was induced when the aqueous phase was changed by a new one. Something that happened every three of four generations. Although the aqueous phase was very easy to prepare, perfectly controlling the pH between batches was impossible, and in general we saw that a few points of difference in the pH made a huge different. It is also important to note that some of the oils we used have a melting point very similar to room temperature. Octanoic acid, for example, has a melting point around 17 C, which can be room temperature in Glasgow. We tried to control as much as we could

the room temperature, but the results described here spanned for nearly a year of experiments, so it is very likely that the conditions during winter or summer were different. Based on all this, we think that the platform was suitable to perform experiments like the ones explained here, but we are not sure at the moment it would be a viable platform to perform more "chemistry pure" experiments, where a much better control of the temperature, noise and contamination is needed.

7

Discussion and future work

The ultimate objective of our research was to create from scratch an autonomous evolvable system. Although it might seem that the easiest way to achieve this is to create it *in silico*, a software system is inherently disembodied, and this severely limits its open-endedness, which is a firm requirement for full autonomy. Thus, we decided to focus on *in vivo* systems, with the requirement that they cannot be autonomous or evolvable before starting the experimentation, therefore these properties must emerge from the system itself. This limits the initial system to non-living matter. Moreover, we must also limit our role to just defining the inputs, after which the system should self-assemble and evolve. We know that this process happened at least once, four billions of years ago during abiogenesis, when life arose from non-living matter. Therefore, by emulating the origin of life we could potentially create a fully autonomous evolvable system. This is easier said than done, because no one has been able to emulate or replicate the origin of life.

Abiogenesis is a huge research field that covers many different topics, and in particular our projects focused on the role of oil-in-water droplets as plausible model for protocell membranes. We wanted to investigate how lipid aggregates could chemically evolve into more complex compartments with life-like behaviour in a similar way to what might happen during abiogenesis. Nevertheless, our research is not situated in the "lipid world" school of thought. We do not consider that life started with lipid compartments, although we think that they had an active role during the origin of life, and that they were not mere passive actors that just encapsulated RNA/DNA. Simple lipid aggregates offered means for a proto-metabolism, and they helped with basic cell functionality like division, signalling, morphogenesis or motion.

The origin of life unlikely happened in a vacuum. There were probably external phenomena that forced molecules to mix, that selected them based on some of their properties, that oscillated them between different conditions as simple as "warm" and "cold", that created out of equilibrium media, etc. Those first simple molecules were not autonomous, and they needed external help. Planet Earth during abiognesis was a huge platform governed by the laws of physics and thermodynamics that interacted with the available chemistry. Based on this, we considered as indispensable the use of an external platform, which coupled with chemistry, could perform an evolutionary process. This is why most of the work of this thesis centred around the design and implementation of such platforms. We do consider the platforms as active members of evolution.

The use of an external platform has a huge drawback though, because it removes the "autonomous" property from the system, which we consider indispensable. This

leaves us with a "chicken-egg" paradox. On one hand, if we add a platform the system is not autonomous, on the other hand, without a platform the system is not evolvable. Evolution needs autonomy, and autonomy needs evolution. Our solution to this problem was to follow a "top down" approach, where initially we created an evolvable system that fully depended on the external platform, and then, in iterative steps, parts of the platform were removed, and the functionalities removed were added into the system, making it more autonomous.

7.1 Dropbot discussion

The first platform we built was a liquid handling robot (LHR), because if we consider how a human would perform a droplet-based experiment, LHRs are the ones that follow a more "humanized" way, replacing the manual steps with actuations.

Because none of the LHRs available in the market were suitable for our experiments, we decided to design and create one from scratch, which we named "Dropbot". Its design was heavily based on a 3D printer, replacing the extruder by a liquid dispensing system. The 3D printer itself was used as a prototyping tool in order to build the robot frame, its translation mechanisms and its dispensing actuations.

This robot was connected to a computer that executed a genetic algorithm, and was equipped with a camera that acted as a sensor and sent back to the computer information about the experiments, so that the evolutionary algorithm could iterate. Our experiments were based on oil-in-water droplets, in particular we focused on the role of surfactant molecules, like fatty acids, because it is known that they were available during prebiotic planet Earth. The objective of this project was to evolve surfactant aggregations into droplets with life-like behaviours, such as movement and division.

As our results showed, we achieved this because random initial formulations would eventually be optimised towards the defined behaviours by using their compositional information as a genome. Not everybody agreed with the implications of our results, and many said that we were just optimizing chemistry, something already done before, and therefore our results were unrelated to abiogenesis. The main difference between our results and *just* chemical optimization is that our experiments happened on two time frames, just like life. One of these time frames was the evolutionary one, which happened in a computer, and performed all the genetic operators and natural selection as required. The other time frame happened within the droplets as entities, living "here and now", following chemotactic trails, exchanging components with the medium, and working to maintain their integrity. Nevertheless, some researchers will never agree with us and will always consider our experiments just chemical optimization, although "just chemical optimization" had to happen during abiogenesis.

During our evolutionary experiments the most interesting results obtained were the discovery that for the division behaviour, the system evolved into two different genotype islands, which were represented by a different phenotype each, while for the movement behaviour, the system evolved into two different genotype islands that were represented by the same phenotype. These results showed that our system was non-linear, and non-linearity is a property of biology.

From a *platform engineering* point of view, our platform was one of the best examples of automating lab operations using 3D printing prototyping, and as far as we know our platform was also the first liquid handling robot built this way. It is interesting how the use of 3D printers for lab automation has exploded and hundreds of projects are being developed nowadays. An example of this is the presence of articles about 3D printing in top journals, like Science, or the quantity of articles in the same line published almost every week in more specialized journals like "Lab on a chip" or the "Journal of Laboratory Automation". 3D printing is here to stay, not only because it is a good and cheap prototyping technique, but because it allows for designs that are impossible with standard prototyping techniques, like injection moulding. It can be said, though, that a lot of the articles published using 3D printers are mostly proofs of concepts, and our work is one of the few ones where the system did work and produced relevant scientific data.

7.2 Flowbot discussion

From a conceptual point of view, the objective of the Flowbot project was to remove elements from Dropbot, following a "top down" approach, in order to obtain a simpler platform, where mechanical actuation was embodied into it. In order to do so, we inspired ourselves in the development of microfluidic and millifluidic devices, and we studied how such a device could encapsulate most of the functionality of a liquid handling robot. We wanted to keep the device's fabrication as simple as possible, and that is why we set the requirement of the device being manufactured in a single step. This had the drawback of making the fabrication more complicated, because 3D-printing big devices is often not as straight-forward as expected, but it had the advantage that we could 3D-print different experimental arenas into the device itself.

Our first objective was to replicate the results obtain in Dropbot within Flowbot, and we achieved it as shown during its own result chapter. Using the same inputs as Dropbot, the evolutionary algorithm could also optimize droplet formulations to a defined fitness function in Flowbot.

Our second objective was to exploit the characteristics of our device to show that by using different 3D-printed environments across an evolutionary experiment, we could alter the evolutionary trajectories. The first result obtained was that the average fitness value through generations, which is generally expected to grow or stay still, plummeted every time there was an environmental variation. This showed that changing the environment did have an effect on evolution. The second result we obtained, after investigating the data, was that the fitness landscapes for each environment had a different shape, although all of them seemed to have a peak in a similar position. The way the shape varied was interesting, because the basic environment which just used an empty container, like a petri dish, had a flat and extended fitness landscape, where the fitness grew steadily to the peak from every point. On the other hand, the other environments we tried had rugged fitness landscapes, with most of it surface returning extremely low fitness values, while only a few of the fitness points were above zero and rose sharply to its peak. Comparing the fitness landscapes, it could be clearly seen that the environments

using obstacles were a subset of the one using an empty arena, which led us to think that the environmental changes were filtering the populations. The third and last result we have extracted was a comparison of the genotype values using a "one-way analysis of variance" (ANOVA), that certified that the environments had an impact on the genotype values. As far as we know, ours is the first system made from scratch that studied the relation between environment and entities *in vivo*.

If we consider how evolutionary theorists study environmental changes, they state that organisms and populations will go through phenotypic plasticity or phenotypic diversity. Some of them might consider the role of epigenetics later on in order to change the genotype, but during a first instance everything happens in the phenotype. Studying our results we did not observe any phenotypic plasticity, and the phenotypic diversity we observed was induced by the evolutionary algorithm, which always applied the same genotypic drift, regardless of the environment changing.

Overall these results showed that we were able to embody some of the evolutionary operators from the evolutionary algorithm into the device itself. Critics may argue that from a conceptual point of view it does not make a difference with respect to the Dropbot project because we still have an artificial platform guiding evolution, but as we said before, evolution and abiogenesis could not happen in a vacuum, so there had to be *something* guiding evolution. We are satisfied with our results because we managed to transform some of the mechanical actuation into passive elements of the system. That said, we are still very far away from the "autonomous evolvable system", and the results showed by the Flowbot project are only a tiny step in this direction.

7.3 Future work

The "future work" of the Dropbot platform was already discussed in detail in its own section (Section 4.6, p.108), where some of the technical components that could be improved or implemented were described. In the same way, a more technical "future work" regarding the Flowbot results was also discussed in its own section (Section 6.7, p.168). We will focus here on how the core idea of how an "autonomous evolvable chemical system" could be pushed forward.

Flowbot is very far away of being an autonomous and evolvable chemical system. The "evolution" it performs depends completely on the platform. Therefore, the question is "what can be removed next?". We think that the next step might require the use of a new platform, the same way we went from Dropbot to Flowbot, and we think that the best one would be a microfluidic device.

Initially we would like to replicate the results obtained here using a microfluidic device, and then in order to embody more functionality into the system, we think that the valve actuation from microfluidic devices are an excellent candidate to encapsulate new functionality, because with valve actuation we would be able to potentially change on-line the flow-rate, closing the loop between outputs and inputs. Microfluidic devices are a more extended and studied platform than millifluidic devices, and we think we could apply a lot of its existing knowledge into our new platform.

Microfluidic devices also allow for "droplet" operations, like droplet fusion, fission, sorting, etc. The use of microfluidic droplet operators could potentially allow us to work directly with droplets instead of ratios and flows. Thus, we could save the best droplets from an experiment, instead of saving its digital recipe, and then we could use them to generate new droplets based on genomic operators. This way, we would take another step forward away from the computer.

One of the main limitations of the Flowbot platform was its number of experiments per hour - roughly 20 different experiments when using two platforms in parallel. A microfluidic device could potentially increase the number of experiments by one order of magnitude, while also reducing by the same order of magnitude the quantity of reactants, solvents and waste used. Not only that, but microfluidic devices already integrate a wide range of different analytical techniques, which would provide us with more data that could be integrated into the evolutionary algorithm. The main problems to solve regarding this platform are, as discussed during the introduction, if we would be able to manufacture them, and how our chemistry will need to change in order to be suitable for the polymers used in microfluidic devices. Finally, microfluidic devices can also be integrated into liquid handling robots in order to increase its number of inputs.

8

Experimental

8.1 Dropbot project

8.1.1 Robot frame implementation

8.1.1.1 Staging area

The staging area consisted of a large glass plate, fixed to the robot frame. This area was defined by being the space over which the X-Y carriage could move and hence all apparatus that needed manipulation by the carriage were placed on the stage. At the centre of the stage there was a 96-well plate, in which fluids were placed for formulation mixing, prior to droplet placement. Each well was equipped with a magnetic bar. There was a magnetic stirrer underneath the well-plate, used to actuate the miniature magnetic stirrer bars inside each well. Also, laying on the stage were two Petri dishes, one was used to carry out the experiments and the other was used to collect waste after experiments were finished. Manual intervention was required after a series of 48, or 96 experiments (depending on whether experiments ran overnight) to clean the well-plate and waste petri dish.

8.1.1.2 X-axis

Figure 8.1 shows a 3D-rendering of the X-axis translation mechanism. The same design was mirrored on both sides of the frame. This mechanism is static in relation to the staging area and frame of the robot.

8.1.1.3 Y-axis

Figure 8.2 shows a 3D rendering of the Y-axis translation mechanism. This mechanism ran along the X-axis via the X-axis belt, rod and linear bearing system. The two round steel bars seen in Figure 8.2 were inserted into the complimentary holes seen in Figure 8.1. A single motor was mounted on one of the X-axis carriages to actuate the central belt and provide linear motion along the Y-axis.

8.1.1.4 Mobile carriage

Figure 8.3 shows a 3D-rendering of the X-Y carriage. This component ran along the Y-axis via the Y-axis belt, rod and linear bearing system. The "ridge" running from left to right along the Y-axis served as an attachment point for the circuitry

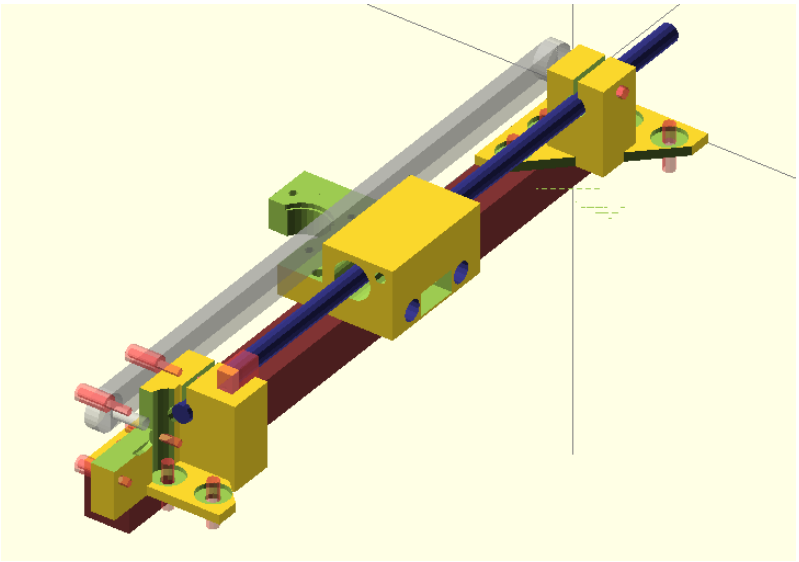


Figure 8.1: X axis 3D design. **Blue**: The precision rod (8h7). **Grey**: The timing belt (T2.5x6mm). **Yellow**: The 3d-printed parts. **Dark red**: Aluminium strut profile (20x20mm). **Transparent red**: The parallelepiped near the motor is where the end stops were placed; other parts in this colour are machine screws used to fasten the parts together.

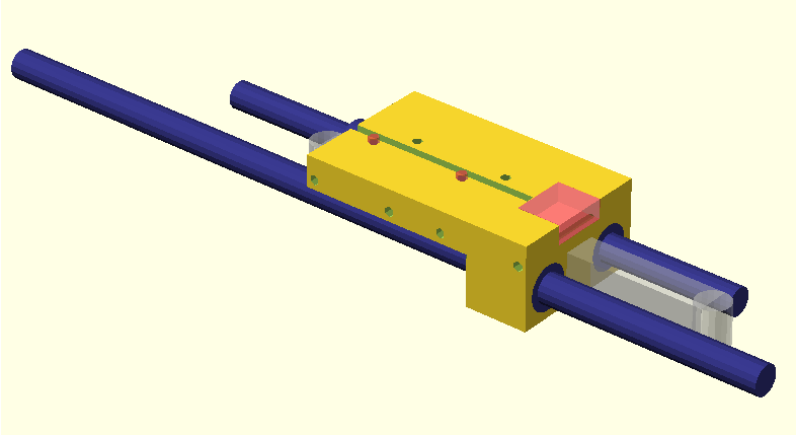


Figure 8.2: Y axis 3D design. **Blue**, the precision rod. **Grey**, the belt. **Yellow** the printed parts. The motor that moved the carriage around the Y axis could be seen on Figure 8.1. The red transparent parallelepiped is where the end stops were placed. The four holes on its side, and on the other side where it cannot be seen, were used to place the structures that would handle the syringes or any other equipment, making the design customizable.

(Section 3.2.1.3, p.67). The design of the syringe actuation mechanism is detailed in Section 3.2.1.2.3 (p.66).

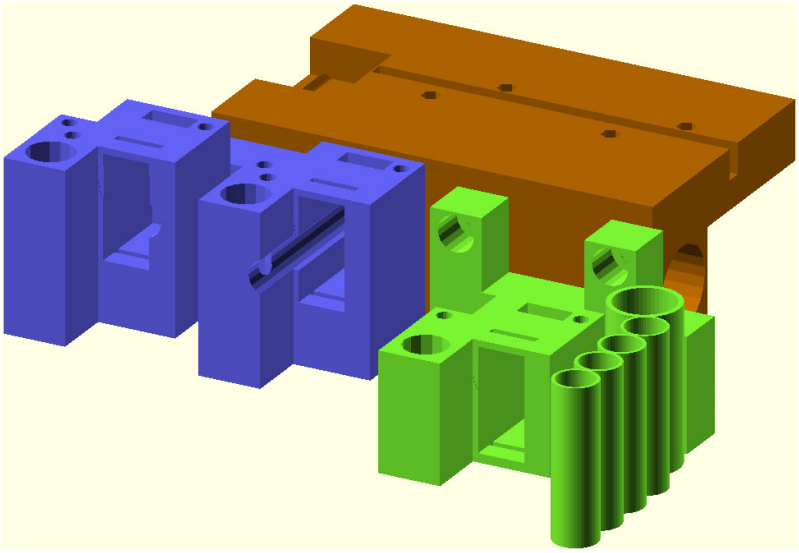


Figure 8.3: The X-Y carriage, assembled but without syringes (see Section 3.2.1.2.3). **Blue:** Locations for actuated syringes. **Green:** Locations for fluidic connections. **Brown:** Attachment to the Y-axis.

8.1.1.5 Fluid platform

The fluid-handling platform was a simple frame, constructed from aluminium strut profile and the 3D-printed assembly pieces used in the main robotic frame. Polycarbonate sheets were attached to the strut-frame to support the weight of the pumps. The frame was designed so that reagents could be placed underneath the pumps. This design was chosen to minimize the effect of any chemical spills by preventing contact with the electronics. The design of the pumps themselves is detailed in Section 3.2.1.2.1 (p.64). The physical platform itself can be seen in Figure 8.4 (p.179).

8.1.2 Bill of materials

- The frame was built using Bosch-Wrexroth 20x20mm aluminium strut profile, fastened together using custom, 3d-printed polylactic acid (PLA) pieces.
- The motors used were NEMA ([NEMA, 1998]) 14 for the Y-axis and two NEMA 17s for the X-axis.
- The linear motion mechanics were derived from the RepRap printer, using a belt (Timing belt T2.5x6mm) and pulley (T2.5 pulley, 5mm bore) system connected to the motors.
- Round-profile hardened-steel bar (8h7, chrome plated) and round-profile linear bearings (LME8UU) were used to achieve smooth linear motion.

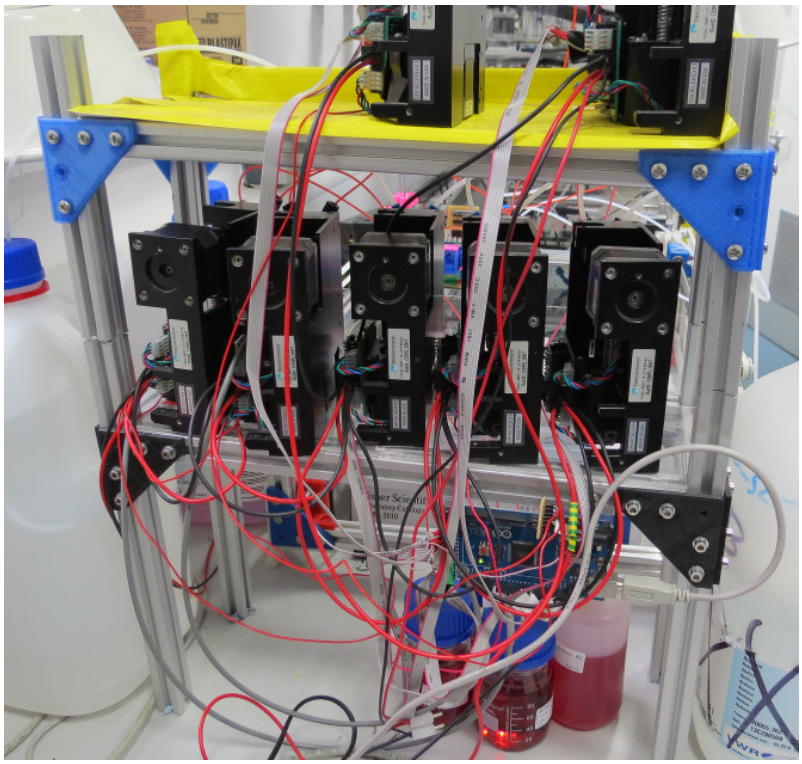


Figure 8.4: The syringe pumps were placed in a platform, external to the robot itself. The reactants were placed at the bottom of this platform.

- The syringe pumps modified and used were “TriContinent C-Series”.
- The syringe pumps valve used were 3-way PEEK.
- Arduino Mega boards were used to control both the linear motion and the syringe pumps.
- “IDEX Health Science PEEK 1/8"” tubing was used to connect those pumps used to introduce or remove cleaning solvent and aqueous phase to/from the petri dish arena.
- “IDEX Health Science FEP Ora 1/16 x 0.20"” was used to carry organic phases from reagent bottles to syringe pumps and from syringe pumps to the X-Y Carriage.
- The syringe used to direct the tubing from the syringe pumps at the carriage the “1 ml NORM-JECT”.
- These syringes were fitted with “Weller KDS16TN25 Needle Taper Tip 16G”.
- The syringe used in the servo-actuated syringes was a “Hamilton 710 LT 100 µl”[1].

[1]We tried syringes with a smaller volume in order to have a better control over the volume displacement, but the problem we had is that syringes smaller than 100 µl used to have very thin plungers that would bend easily.

- The needles used were "Weller KDS2012P Dispensing Needle GA20 ID 0.66 MM".
- The syringe was raised and lowered by a "New Power XL-3.7" servo-motor.
- The plunger was actuated with a "9g servo motor". In this specific case, a "TowerPro Micro Servo 9g SG90".
- The motor arm used was the default arm, which shipped with the servo motor.
- The well plate used was "Nunc U96 PP 0.5 ml".
- Inside every well, a "Magnetic stir bar micro PTFE 6 mm x 3 mm" was used to provide liquid turbulence.
- Below the well plate, was a single "Variomag Compact" stirrer plate, used to turn the stirrer bars.
- A PS3 EyeToy camera was used to record the experiments.

8.1.3 Platform safety

The platform itself did not have active safety mechanism, and it was designed to not fail (passive safety). We tried to completely isolate the electronics from the chemistry, and the platform was always placed inside a fume hood. All the reactants, solvents and different phases were placed inside safe containers, and all the waste was properly disposed. The power supply used a safety socket when plugged in. The robot never ran outside lab hours, and it was checked at least once every hour.

During the two years of operation it had two failures. One of them, when we were still assembling it and the platform was not inside a fume hood because it was not handling chemistry yet, happened because one of the PCBs had a short circuit.

The other failure happened at the end of the experimental phase. The crank shaft that lifted or lowered the syringe became detached, meaning that the syringe was in low permanent position, and while it moved around it pushed equipment around.

8.1.4 Chemistry[2]

The oils and surfactants were purchased from Sigma-Aldrich and used as received, unless otherwise stated. All the experiments were performed at room temperature, and the Petri dish used was made of glass and had an inner diameter of 4 cm.

Figure 8.5 shows the chemical structure of the different molecules used during this research. Dodecane, 1-octanol, 1-pentanol. diethyl phthalate and octanoic acid were used in the oil phases, while tetradecyltrimethylammonium bromide (TTAB), was used in the aqueous phase.

Table 8.1 details some of their most important properties.

[2] The chemistry of this project was designed by James Taylor. Although its design is not part of this thesis, it will be described here in order to cover all the different topics of this project. James was the person who designed it, but I was the person carrying the experiments, so part of my work was to prepare the solutions and reagents as described.

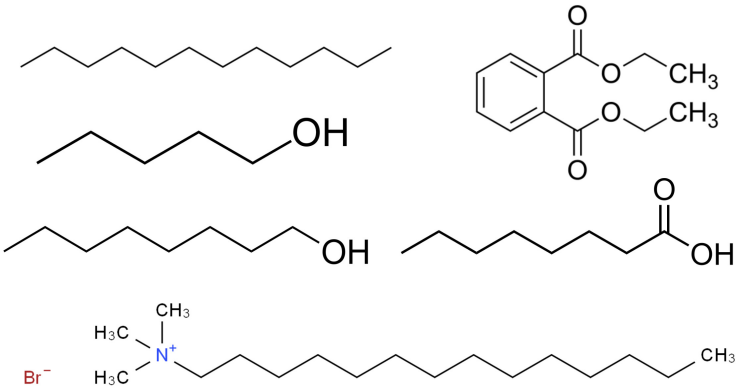


Figure 8.5: Chemical structure of the molecules used in the Dropbot project. From top left, and going clockwise: dodecane, diethyl phthalate (DEP), octanoic acid, tetradecyltrimethylammonium bromide (TTAB), 1-octanol, 1-pentanol and dodecane.

Oil phase	Density (g/ml)	Solubility (g/L)	Surfactant?
Dodecane	0.75	Insoluble	No
DEP	1.19	1.08	No
Octanoic Acid	0.91	0.68	Nonionic
(Sodium octanoate)	n/a	50	Anionic
1-Octanol	0.824	4.6	Nonionic
1-Pentanol	0.811	22	Nonionic
Water phase			
TTAB (pH 13)	n/a	10% (w/v)	Cationic

Table 8.1: [Data from Sigma Aldrich] Main properties of the phases used during our experiments. It is important to note that Sodium octanoate was not part of our inputs, but it was very likely to appear when octanoic acid was placed into basic water. Our experiments contained around 20 mg of oil phase per 2 ml of aqueous phase (or 20 g per litre), therefore, although the solubilities of the oil phases might seem low, they were very important in our system. Pentanol droplets, for example, would completely dissolve.

8.1.4.1 Oils and aqueous phases preparation

TTAB (6.73 g, 20.0 mmol) was dissolved in distilled water (*ca* 600 mL), adjusted to pH 13.00 with 5 M NaOH solution and made up to 1 L to give a 20 mM solution at pH 13.00.

The oils 1-octanol, octanoic acid, dodecane, 1-pentanol and DEP were prepared in 200 mL aliquots in reagent bottles. Each oil was dyed with 0.25 mg/mL Sudan III and vortexed to mix. In the case of octanoic acid, it was both tested pure, but also in 20/80 ratios with 1-octanol, dodecane and DEP where octanoic was present at 20% and each of the other oils an 80%.

8.1.4.2 Cleaning cycle

After each experiment, acetone (*ca* 3 mL) was pumped into the Petri dish to dissolve remaining oil droplets and the mixture was aspirated from the dish to the waste container. The dish was then washed with acetone (2 x 3 ml) and aqueous phase (2 x 3 ml).

8.1.5 Experimental protocol

A fully automated liquid handling robot capable of producing droplets in a Petri dish with an aqueous sub-phase was constructed. The robot was based upon a RepRap 3D printer architecture with a webcam for video recording / image analysis.

The droplet formulations were produced in well-plates (96-well format) where each well was stirred with a magnetic stirrer bar. The droplet formulations were based upon the following reagents ((1-octanol, 1-pentanol, diethyl phthalate (DEP), dodecane and dilutions of octanoic acid in one of the other oils (typically 20% v/v)). The formulations were delivered from the computer control in the form of four numbers, which represented the proportion of each oil; therefore the total summed to "1.0". The total volume of the oils place in a well was 360 μL. The specific quantity for each oil was this volume multiplied by their relative proportion.

Once the mixing was completed, three experiments were conducted from the final formulation. The Petri dish was initially filled with 2.2 ml of aqueous phase. For each experiment, 80 µl of oil were absorbed using the syringe. Of this volume, four 5 µl oil droplets were then placed in the aqueous phase at different positions. These positions were consistent between experiments, with pre-programmed X-Y coordinates. In order to generate droplets the needle was laid just above the aqueous phase and the droplets were released to the surface of the liquid. Once a droplet was outside the needle, the syringe was moved up to its default position. This movement broke the tension between the droplet and the needle, depositing the droplet over the aqueous phase[3]. This process is summarised in Figures 8.6 and 8.7.

A video of the resulting droplet behaviour was then recorded from beneath the dish using a camera at a resolution of 640 x 480 pixels at 30 FPS after covering the top of the Petri dish with a white background for 60 seconds. Image analysis of the droplet behaviour was used as fitness function for the evolutionary algorithm.

[3]Although not reported, we saw a big difference in behaviour when the droplets were placed inside the aqueous phase instead of over it

After a complete GA generation concluded, the evolutionary algorithm quantified the fitness of the formulations and calculated the next set of reagent volumes.

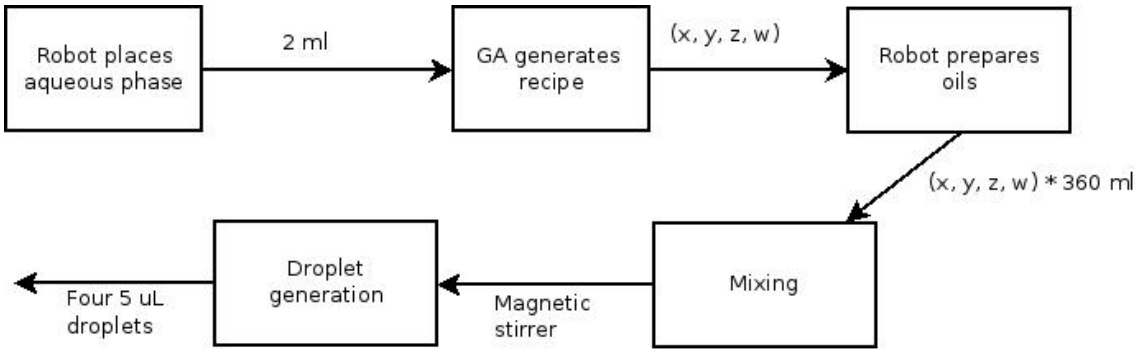


Figure 8.6: Outline of the droplet generation. First the aqueous phase was placed, and then droplets were placed over its surface.

After each experiment concluded, the syringe pumps were used to pump acetone and aqueous phase into the Petri dish and then to remove the solvated waste. First, three washes of acetone were performed: 4 ml, 4 ml and 3 ml. During the first two of these washes, the needle was dropped into the acetone, and the plunger was actuated to absorb and release acetone, cleaning the needle and syringe. After washing with acetone, three washes with aqueous phase were performed, with 1.5 ml, 1.5 ml and 1 ml. The objective of the aqueous phase wash was to remove the remains of acetone from the petri dish so that it would not interfere with subsequent experiments. This process is summarized in Figure 8.8.

8.2 Flowbot project

8.2.1 Platform description

A fully automated fluidic device capable of producing droplets in a Petri dish like arena with aqueous subphase was constructed. The device was a monolithic which had a series of inputs and a outlet connected to liquid pumps. The syringe pumps used 500 µl syringes for the four oil phases, and 5 ml syringes for all the other pumps. The pumps used 3 way PEEK valves, as provided by Tricontinent. FEP tubing was used to connect all the liquid components. The pumps used were defective "Tricontinent C-Series Syringe Pumps". Their stepper motors were connected to "Pololu a4988" drivers. The drivers were controlled using an "Arduino Due" board. A homemade PCB shield was used to connect the stepper drivers to the Arduino board, and custom firmware was written using the Arduino suite in order to control the pumps. Labview software was used to control the experimental protocol, and a Python library was written to analyse the experiments using visual information. Above the arena there was a camera for video recording/image analysis at about 4 cm.

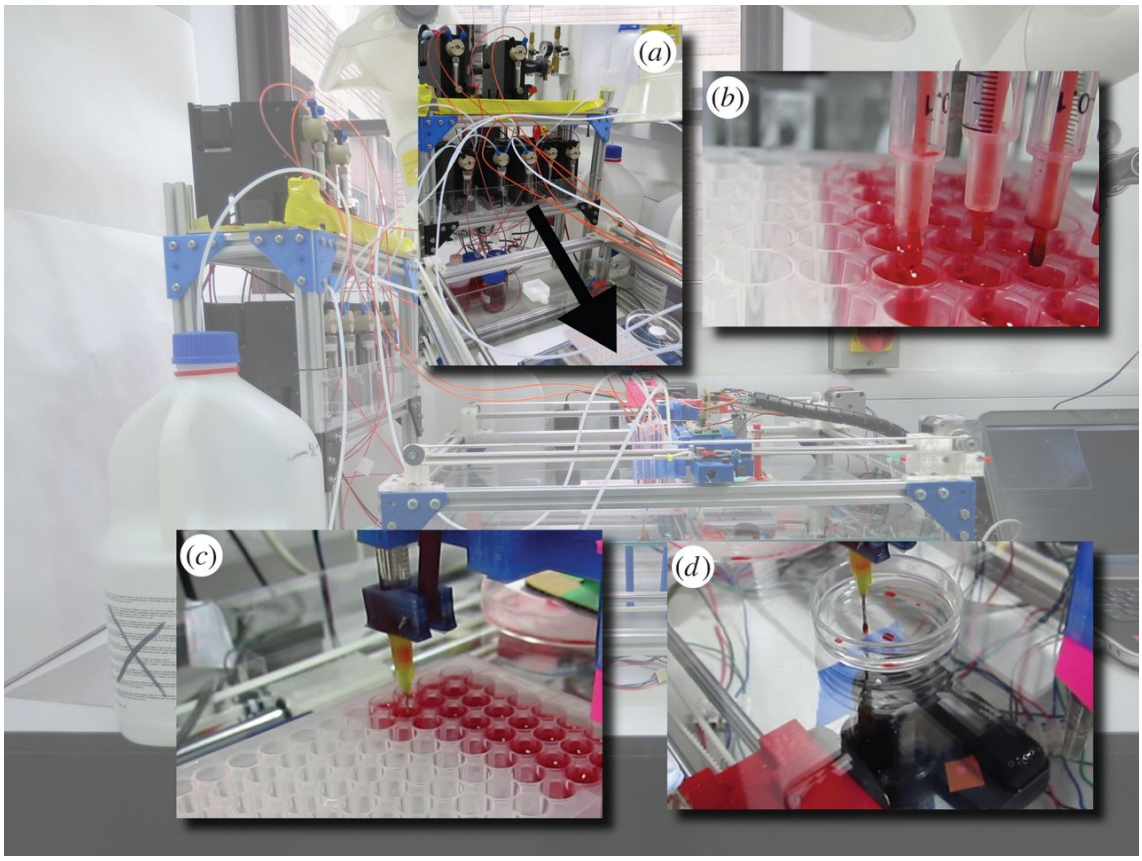


Figure 8.7: The robot can be seen in the background, with the liquid pumps and reactant bottles to the left, the working space with the liquid-handling carriage in the centre, and the electronics and computer to the right. The liquid pumps (a) and the moving carriage were used to place the reactants into the 96-array well plate (b) after positioning the outlet above each desired well. Each well contained a magnetic stirrer to mix the formulation. The carriage (c) was also equipped with the described automated syringe, which took up the mixture from the designated well, and then generated the set of droplets in the Petri dish (d). Once this process was finished, the camera would record everything that happened during the experiment.

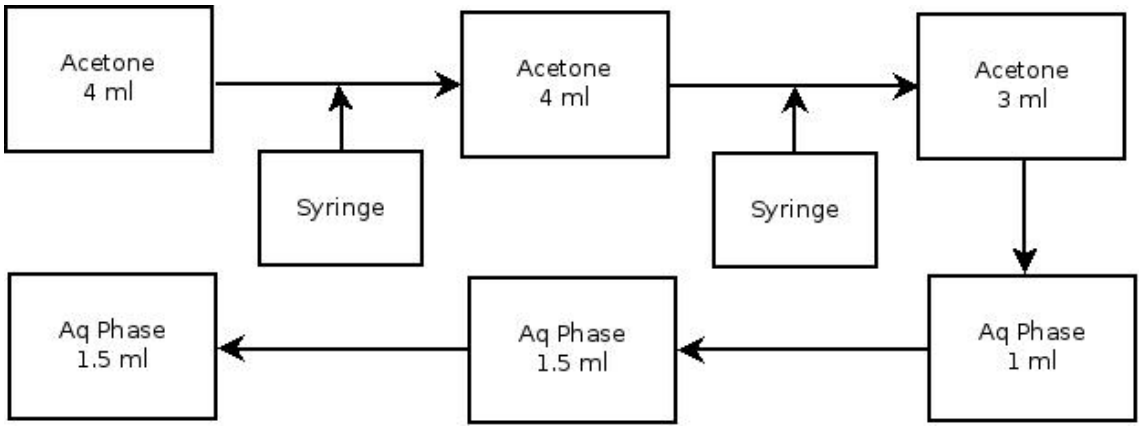


Figure 8.8: Outline of the cleaning procedure. At first, three washes with acetone were performed. Between the first and second, and the second and third, the syringe is dipped into the acetone, and the plunger actuated, cleaning the needle, barrel and plunger. The three following aqueous phase washes were performed to remove the remains of acetone.

8.2.2 Bill of materials

- Each device was designed using Rhinoceros CAD software, and 3D printed using the 3D printer "Bit from Bytes" using PP as thermopolymer.
- The syringe pumps used were "TriContinent C-Series". Three of them used 5 ml syringes, and four of them used 500 µl syringes. All of them used the same 3-way PEEK valves.
- The electronics from these pumps were replaced with custom-made PCBs in order to power them using an Arduino board.
- The stepper driver used to power the syringe pumps were Pololu a4988.
- An Arduino Due was used to control the syringe pumps.
- "IDEX Health Science PEEK 1/8"" tubing was used to connect acetone, aqueous phase and waste from the containers to the syringe pumps, and from the pumps to the device.
- Flangeless fitting nuts, 1/8" OD Tubing, PEEK, were used to connect these tubes to the syringe pumps and device.
- "IDEX Health Science FEP Ora 1/16 x 0.20"" was used to carry organic phases from reagent bottles to syringe pumps and from syringe pumps to the device.
- Flangeless fitting nuts, 1/16" OD Tubing, PEEK, were used to connect these tubes to the syringe pumps and device, with corresponding cone shaped fitting.
- A Microsoft LifeCam was used to record the experiments.

8.2.3 Manufacturing the device

The process that goes from a 3D model to a physical 3D printed device can be divided into four steps:

1. Choosing a filament is probably the most important step, because different polymers have different characteristics. For example, the most commonly used polymer, polylactic acid (PLA) is very easy to print, but it is biodegradable, and it is degraded by organic chemicals, like acids or solvents.

2. Starting with a STL (STereoLithography) file which describes the 3D model from a geometric perspective, "slicing" software is used to transform the STL file into a G-Code file. G-Code files describe the mechanical operations the 3D printer will need to do in order to manufacture the piece. The user will need to configure[4] the slicing software in order to produce the desired output.

3. Configure the 3D printer itself. This involves setting the hot end temperature, which is directly related to the chosen polymer, and some minor mechanical adjustments, like initial layer height (also known as "Z-distance").

4. Input the G-code into the 3D printer, and let it manufacture it.

These four steps will now be described for the particularities required to print one of the devices described in this chapter.

Initially we will consider the case where everything worked fine, and the 3D printer produced a perfectly viable device. Unfortunately this was rarely the case, and in the experimental chapter we will cover some of the most common problems we had when trying to manufacture these devices (Section 8.2.4).

8.2.3.1 About the filament

The filament used to 3D print our devices was polypropylene (PP), with a diameter of 3 mm as required by our 3D printer. PP was chosen for its very high chemical resistance, which makes it a perfect candidate to be used with chemistry. This is not the case with the most widely used plastics, like PLA or acrylonitrile butadiene styrene (ABS), because they degrade very fast when exposed to, for example, organic solvents like acetone. As far as we know, we are the only research group printing with this filament, and the reason why it is rarely used is because it is very difficult to use. Thankfully, the group had already mastered its use, so we did not have to develop the protocol in order to use it in this research. In any case, its particularities are going to be described now in case the process is to be reproduced.

The main complication of 3D-printing with PP is that none of the commonly used printing beds[5] worked with it, because PP was not sticking to them. Neither glass, Kapton Tape or any of the usual materials worked when heated or when left at room temperature. The only material PP would stick to was itself. Therefore we

[4]Layer height is the most important one. Another important configurable paramaters are the filling ratio, or the filling pattern.

[5]Like the ones used when 3D-printing PLA or ABS.

used a PP sheet as printing bed. The problem this creates is that after a piece has been printed it needs to be removed, while when printing with PP over a sheet of the same material the piece would fuse into it. The way we solved it was to print first a raft at a slightly lower temperature (230 °C)[6], and then the rest of the piece at 260 °C, which is PP's optimal printing temperature. The extrusion at 230 °C was not perfect, but it was good enough to print a raft. Then, the whole piece could be peeled off, and the bed reused for a new piece.

8.2.3.2 About the 3D printer

In this research the 3D printer used was a "3D Touch" from "Bit from Bytes". It was a standard "fused deposition modelling" (FDM) 3D printer, which could operate three different extruders, although we only used one.

This 3D printer had two interesting properties, which were not unique to it, but which helped to manufacture our devices. On one hand, its printing area was very big, around 35 cm x 20 cm. Commonly FDM printers have an area of 20 cm x 20 cm, although obviously there are also several models with bigger printing areas. This was important, because checking back Figure 5.9, it can be seen that the base area of our device was 14 cm x 9.6 cm. In theory this would perfectly fit in a printing area of 20 cm x 20 cm, but when 3D printing a piece it is needed a safety perimeter which in this case would be very near the 20 cm. The other interesting property was that the extruder only used metal parts, because most extruders use a PTFE tube to guide the filament into the hot end in order to minimize the friction between the filament and the extruder. PTFE degrades at around 240 °C, and although that is not a problem when working with PLA or ABS, PP's ideal melting point is around 260 °C. Metal hot ends are not unique to the 3D printer we were using, and nowadays they are probably starting to be the most common option, but it was a good starting point, because they can go up to 300 °C when using a thermistor ([RepRap, 2016]), and even higher when using a suitable thermocouple.

Some of the devices were also printed using a "RepRap Prusa i3" 3D-printer with a full metal hot end. "RepRap" printers are the most popular open source option. Similar devices were also printed successfully using an "Airwolf" 3D printer. Any FDM 3D printer with a printing area big enough and a full metal hot end would work. Nevertheless, we decided to move forward using the "3D Touch" because the group had already printed quite a lot of chemical devices using them, and the printing protocol was already well defined.

Regarding the 3D printer itself, nothing was modified, and the devices were printed using its standard hardware configuration. The only variable which needed to be set-up was the Z distance, which in the context of 3D printing means the initial distance between the nozzle and the printing bed. The initial Z-distance needed to be changed in order to accommodate the PP sheet that would be used as printing bed. In our case this sheet had a thickness of 12 mm, but any other sheet with a similar thickness might work[7]. Then, this distance needed to be minimized in order

[6]This value was discovered by trial and error, trying to find a temperature where the plastic would extrude but in a not perfect way, so that the layer would not perfectly stick.

[7]Thinner PP sheets tend to warp too much.

to guarantee that its initial distance would be as close as possible to the layer height (see p.188, Section 8.2.3.3).

The first layer printed was the raft, and in our experience a good raft looked over extruded, without any gap between the lines of filament printed. By minimizing the Z distance as much as possible, when the printer placed a line of filament, this line instead of having a width of 0.5 mm as defined by the nozzle, it would expand to the sides, using a bigger area, see Figure 8.9.

Another consequence of minimizing this distance is that vertical layers of filament would be printed fused into each other, meaning that the device would be less likely to leak.

In our experience, using a correct value for the Z-distance was the most important parameter in the whole process in order to print pieces that would actually be printed, and be functional.

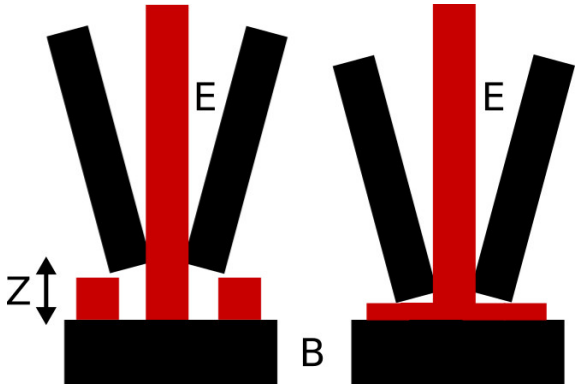


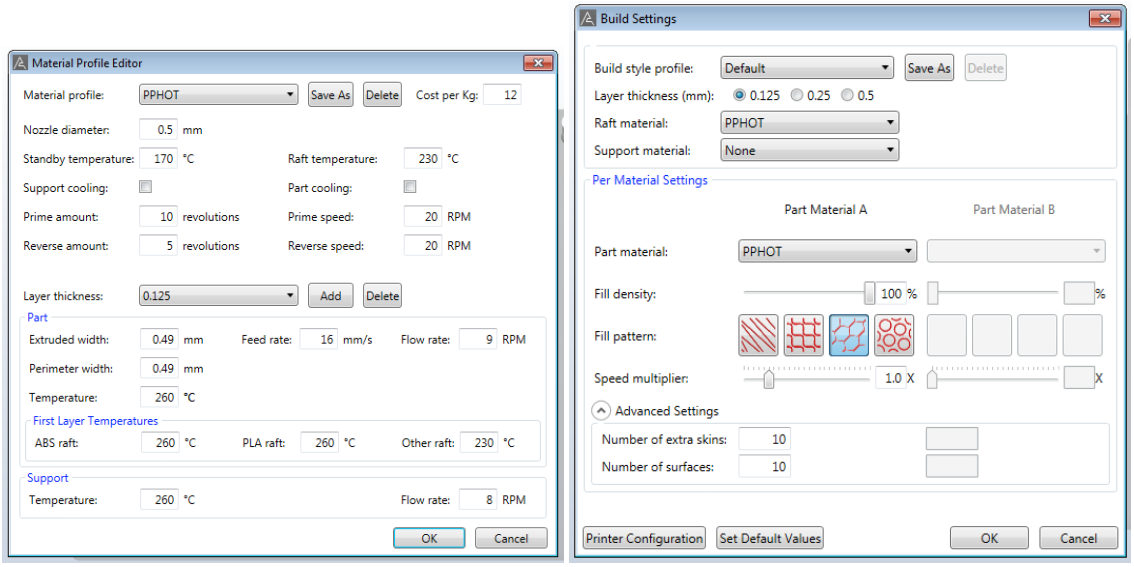
Figure 8.9: **Z:** Z-distance (initial distance between extruder and printing bed). **E:** Extruder. **B:** Printing bed. **Left:** The distance between the extruder and the printing bed is long enough so that the printer can place lines with the defined nozzle diameter. **Right:** By reducing the Z distance, once the filament is extruded, the nozzle pushes it into the printing bed. This way, the filament placed loses resolution, but increases its area and attachment to the base.

8.2.3.3 Axon2 software configuration

The designs described in Section 5.2 were exported into STL (STereoLithography) format using CAD software[8], and then transformed into G-Code using the software Axon2.

Axon2 is a proprietary software developed by Bits From Bytes in order to be used with the 3D printer "3D Touch". If the reader wants to reproduce the work, it is important to use version 2 of the software, because there is a version 3 but they removed some of the features which are necessary to print with PP. There are other programs that can be used to transform STL files into "3D Touch" G-code compatible files, but we decided to stick with Axon2 because it is the one the group used before successfully.

[8]Almost any CAD software will export to STL.



(a) Material profile editor. **(b)** Build settings.

Figure 8.10: **(a)** A new material must be created in order to print with PP following the settings here described. **(b)** Using the material described in (a), the only setting really mandatory is to set the filling density to 100%.

A very quick note about Axon2: Sometimes when loading a STL file, especially if the design is complicated, a pop-up windows will appear saying that there was an error in the design, but that Axon2 can fix it. This happens when not all the surfaces of the design are closed, because for example there is a hanging structure. Axon2 can theoretically fix it, but we recommend to go back to the CAD software and revise it instead.

Figure 8.10 shows the configuration needed in order to print PP devices. We will explain now some of the main configuration parameters.

Axon2 does not include a profile for PP, so the first step is to create one. We named it "PPHOT", but it can have any name. Checking the window titled "Material Profile Editor" most of the parameters are independent of the material or not used at all. The most important are:

(a) Raft temperature. This is the temperature the raft will be printed at. The raft are 3 layers which are printed as a base below the piece itself. As said before, it is important to keep this temperature to 230 °C. If the piece is not sticking to the base during the whole process, this temperature can be raised, but hotter temperatures might make subsequent removal difficult or even impossible.

(b) Part temperature. This is the temperature that will be used to print the whole piece. 260 °C is the optimal temperature in our experience, and it is also the limit top temperature allowed by Axon2. If using another printer, depending on the exact structure of the hot end, a slightly higher temperature might be needed.

(c) Layer thickness. We always used 0.125 mm, because it gives more resolution and because it makes the pieces in our experiment more liquid tight. We have not really experienced with thicker layers, but it is a possibility that can reduce greatly the printing time.

(d) Part flow rate. This is the most important parameter, and while all the other variables were always constant through all the process, this is the only one we played with. Flow rate means the quantity of plastic that is pushed into the hot end, in revolutions per minute. In this case we used 9 RPM, which used to work for us. The higher this number is, the more plastic per minute used, which means the piece will be less prone to leak. On the other hand, the higher this number is, the less resolution the piece will have, the higher the risk of an internal channel being blocked, and also the more the piece will warp during the printing process. This value should be kept at its lowest possible working value.

In the "Build Settings" window, PP need to be chosen as raft and part material with a layer thickness of 0.125 mm. The fill density is set to 100% in order to minimize possible leaks. The most important values here are the number of extra skins, and the number of extra surfaces, which in both cases we set to 10, because that's the highest number allowed by Axon2. The number of surfaces is the number of solid layers below and above an empty channel, while the number of skins is the number of horizontal layers surrounding a channel. These numbers were maximized in order to reduce potential leakage.

Once one of our designs had been build using this configuration, the software would output a "bfb" file, which is just a G-code file generated by Axon2. We might need to open it with a text editor and set the first "Checksum" line to "no", otherwise depending on the firmware the 3D printer might not print it and return a checksum error.

Finally, there is a very last "hack" we did to get better printed devices. As said, the raft was printed at 230 °C, and the piece was printed at 260 °C. The printer will initially print three layers of raft at 230 °C, as expected, but then it will also print the very first layer of the device at 230 °C. It does this to make it easier to separate the piece from the raft, but in our case we actually wanted to keep raft and piece together. It happened to us a few times during the printing process that while the raft would stick to the bed, the piece would peel off from the first layer, and the piece would often leak from this point.

That is why we recommend to open the generated "bfb" file with a text editor, and search for lines that start with "M104", which is the G-code command to set the temperature. The very first appearance of "M104" sets the raft temperature to 230 °C, but we can modify the next appearance of this command to 260 °C, so that the very first layer of the device prints at 260 °C. This way the piece itself will fuse into the raft, and it will never peel off. It will also minimise the possible leaks.

After all this we can input the file into the 3D printer and let it print it. Our devices took around 24 hours to print. If everything seemed OK after 30 minutes, it could be assumed that the piece would print successfully.

8.2.4 Manufacturing troubleshooting

There are several things that can go wrong while manufacturing a device. We will divide the problems into the ones that can happen while printing the device, and the ones that can happen while testing the device flowing liquid through it.

8.2.4.1 Problems while printing

The main problem that might happen while printing is that a device could warp. Warping is a general problem with "fused deposition modelling" (FDM) 3D-printers. Its effects are usually bearable when using PLA or ABS, but in our experience warping was more accentuated with PP. It usually started warping from a corner, and the piece would peel off very slowly. If we were lucky, the warping would not be enough to detach the piece from the printing bed, and it will only cause minor cosmetic effects. If we were unlucky, the piece would completely detach from the printing bed, and it would either move around and fail completely, or get attach to the hot end and potentially damage it[9].

Polypropylene is a plastic with a very high warping coefficient when manufactured, either molded or 3D printed. The technical reason for its warping is because it is a partially crystalline thermoplastic. Its density depends upon the crystallinity and the crystallinity depends on the cooling rate. Depending on the rate at which the piece cools down, it will have a higher or lower crystallinity, which will have an impact on its density - higher crystallinity meaning higher density. When molded, the cooling rate can be uniformly controlled across all the piece using a thermal chamber. The use of thermal chambers in FDM printers has rarely been successful because the thermopolymers used as filament require to be at room temperature everywhere except inside the hot end nozzle. Otherwise they can get a slightly soft, the extrusion would not be perfect and the extruder will very likely get jammed.

If we consider a standard FDM printer printing PP at 260 °C, the moment the filament is extruded it will cool down to room temperature very fast. This is actually good because a fast cool down means it will create smaller crystals, which means it will shrink less. The problem is that the hot end at 260 °C is still moving around, extruding more plastic, and depending on the paths it follows it will be heating some parts more than others. Thus, the piece won't cool down in a uniform way, quite the opposite, and different sizes of crystals will form, with different densities and shrinkage factors. If this is the case, the piece is likely to warp.

Apart from using a thermal chamber, which as said creates a new set of problems, we could optimise the extruding pathing to guarantee a more uniform heat distribution. However, that would not be a trivial solution. In order to try something easier we decided to print the piece over a PP bed, because the first filaments placed will almost fuse with the bed, therefore it won't be able to warp, because it would not be physically able to do it. This process was not perfect, and sometimes the piece warped in a bearable way, while other times it warped so much that the piece was be completely distorted, or worse, it would peel off from the bed, moving around and potentially damaging the extruder.

If the whole piece was warping in a way that the raft was detaching from the printing bed, there were several things that could be done to fix it. The first, and often most useful step, was to reduce the Z-distance until the raft looked solid and over extruded. This step was enough almost in every case. A second possibility consisted of reducing the flow-rate parameter as discussed above. Usually a higher flow rate was related to more warping. Another possibility was to increase the raft

[9]One of our hot ends was damaged this way.

temperature because it would increase the raft attachment to the bed. However, in this case the attachment could be so strong that it made the piece impossible to remove.

If the raft stuck to the bed, but the piece was detaching from the raft, it was necessary to modify the G-code file as explained before. This step should be enough if using Axon2. If some other software is used, often they leave an "air gap" between raft and piece. This air gap must be set to zero, and we recommend to check the G-code generated to make sure that no extra space is left between raft and piece[10].

Another cause for general warping was created when the bed was not perfectly flat. This happened because even though a piece did not seem to warp while printing, it actually pulled the bed and warped it instead. Then, when we removed the piece and tried to print again using the same bed, the Z-distance could vary at different points within the same bed. The best way to avoid it was to simply have a few beds, and once one had been used, let it return to its flat state. Another way to minimise bed warping consisted of baking in an oven the PP sheets with a heavy structure on top.

Another cause for the bed not being perfectly flat happened after it had been used several times, because every time the first layer of the raft was printed the hot end would almost push against the bed, and an area around the center would slowly generate a small dip, see Figure 8.11. In our experience, after one year of continuously printing using the same bed a small valley of around 1 mm appeared, and this caused problems with raft and layer adherence around that area. The best way to avoid this is to print the piece in another position, or to use a new PP sheet.

Figure 8.11: Example of PP bed where a valley was generated after it was used several times. This small 1 mm dip can cause problems with raft adherence and with the device being prone to leak.

[10]Early 2015 we tried to generate the G-code for our pieces using Slic3r or Cura. In both cases it left some sort of air gap that we could not remove. We have not tested if this has been in newer versions of the software.

8.2.4.2 Problems while testing

Once a device had been printed and it appeared to be good, the next step consisted on connecting it to a pumping system and flow liquid through it to check that it was not leaking. A device might leak caused by two opposite effects: it might be under extruded or over extruded. Figure 8.12 shows examples of leakage. If we were lucky the leakage happened very early in the testing stages. Worst case scenario it became apparent a few weeks later when we were already in the experimental phase. During this research only once it happened that a device leaked after a few weeks of use. Therefore, if a device worked during the first days, it was almost guaranteed that it would always work.

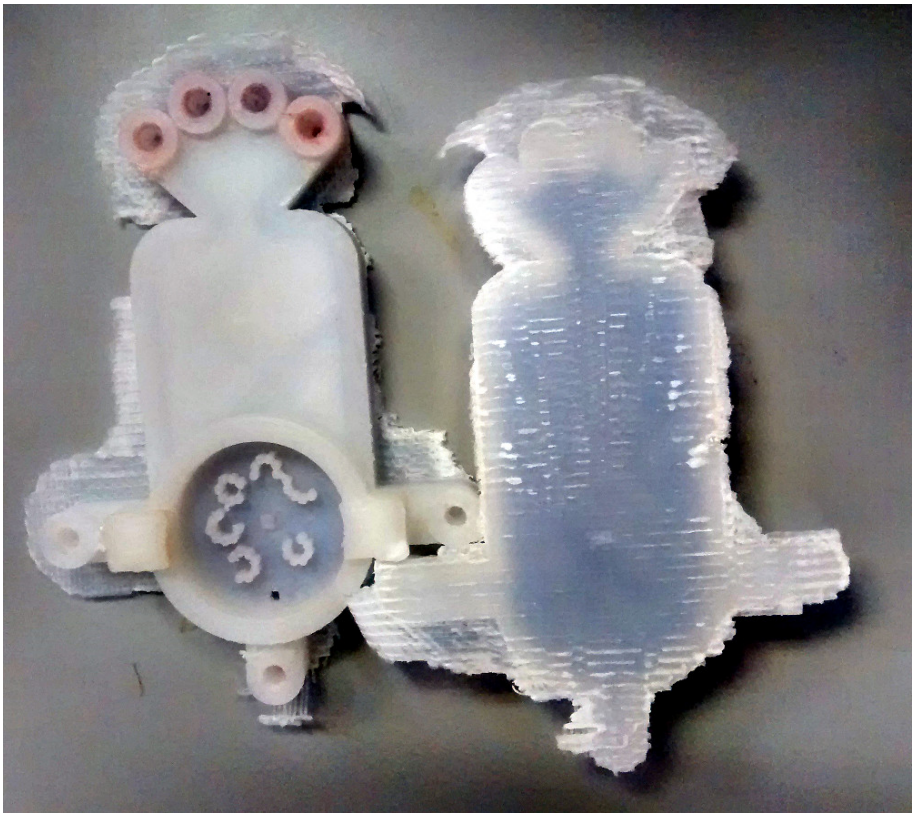


Figure 8.12: Example of leaking devices. In a perfectly working device it would be impossible to see the liquid inside. The only exception was when using translucent filament, in which case we might be able to see the liquid going through its channels.

If the device was over extruded it could leak because some of the internal channels were blocked. This could generate a lot of back pressure and potentially damage the syringe pumps, although usually the fluid just found another way inside the device, and eventually leaked. There were two different strategies to fix it: decrease the flow-rate or increase the Z-distance. If with the chosen flow-rate we had already printed working devices, the best way to procede was to increase the Z-distance, because modifying the flow-rate required the G-code to be re-generated, which could take a few hours. If we had not yet printed any working device, decreasing the flow-rate by 0.5 RPM would be recommended until a working device was obtained.

Apparent over extrusion was not always a bad sign, because if the internal channels were not blocked, the device would be very unlikely to leak. For example, if the outlet used to generate droplets was blocked we would drill a hole using a 0.3 mm drill set instead of discarding it. With over extrusion, it was always best to test it before deciding.

If after trying a few devices all of them leaked from the same point, this was because (a) the bed was not flat (b) our design was wrong. Sometimes it was better to redesign the device instead of iterating with the 3D printer.

If the device was under extruded it could leak because the layers of filament were not fused together. The way to fix it was to decrease the Z-distance or increase the flow-rate. Our recommendation is to firstly decrease the Z-distance as much as possible. Even if the distance between the bed and the nozzle appeared to be non-existent, as long as we could see plastic being extruded it would be good. If reducing the Z distance did not work, then the next step was to increase the flow-rate by 0.5 RPM.

Apparent under extrusion is usually a bad sign. If gaps between the structures were visible, it was better to discard the device, because the likelihood of it leaking was very high.

8.2.5 Droplet generation calibration

Although the devices were always printed the same way, the final result was always slightly different. In almost every case the difference in features between devices was not important, but in the case of the droplet generator outlet, it was, because a few hundreds of microns of difference could mean that the droplets were generated in a complete different way. See Figure 8.13.

Figure 8.13: Flowbot outlet holes differences. Although both devices were printed under the same conditions, differences can be seen in the size of the outlet hole.

Because it was impossible to control how the actual hole was going to be 3D

printed, and because it was not a good idea to apply some kind of manual manufacturing, it was decided to calibrate it using software - the middleware layer.

For each pump we could control the speed at which the plunger would move up or down. Moving the plunger up was here the important bit, because that was when liquid was pushed out into the device, and eventually through the droplet generator outlet. Playing with the speed at which the plunger moved to release liquid, and calibrating it for every particular device we could get each of them to generate droplets in a uniform way.

In particular, we calibrated them for the generation of only 1-octanol droplets, because these were the most difficult to generate (see Figure 8.14). The value for the speed was usually between 50 ns and 100 ns. The objective was to generate around five of them in a row in a consistent way. Thus, although each device had a different outlet hole, we compensated it using a different speed for each of them.

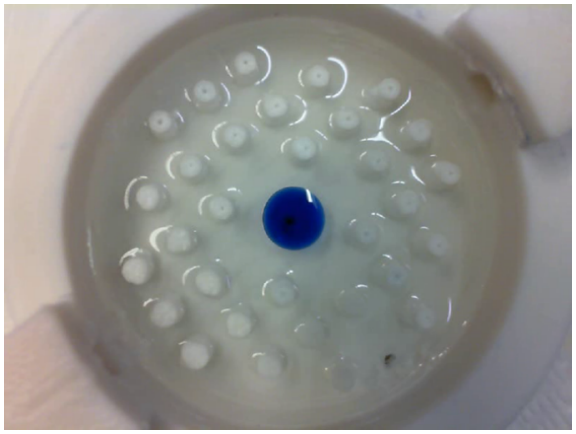


Figure 8.14: Octanol droplets generation failure. Octanol droplets were the most difficult to generate, because instead of releasing into the aqueous phase they used to attach to the outlet and accumulate there. It was important to generate fast pulses in order to push them into the aqueous phase.

8.2.6 Lattice search

Combinations of the four oils individually, in pairs, threes or fours with a granularity of 10% were tested in order to execute the lattice search. In the case of the oils being tested individually, only one experiment was performed, where one of the oils was active, and all the other ones were inactive. In all the other cases, a sequential and evenly spaced search was performed, were every step represented a 10% of the presence of a component into the mixture. This way, for example, in the case of two oils, there would be 9 possible combinations (ignoring the extremes were only one of the oils is active), like: 0.1/0.9 – 0.2/0.8 – 0.3/0.7 – 0.4/0.6 – 0.5/0.5 – 0.6/0.4 – 0.7/0.3 – 0.8/0.2 – 0.9/0.1. The same procedure was applied to combinations of three or four oils. In total there were 11 different combinations, six for pairs, four for triples and one for fours. Therefore, our lattice search consisted of 282 different oil formulations. Each formulation was tested 5 times, generating 1410 videos.

8.2.7 Database

During the previous project the database management was done using the "MongoDB" library. It was our intention to implement something similar for this project, probably using "SQLite" because it works by default with Labview, but we did not have the time to do it.

Therefore, the database implemented for this project was just a text file, where every time an experiment was executed a text line was written with its recipe and the fitness value. If we have had the time we would have implemented a real database system.

8.2.8 Platform safety

All the elements were designed and tested individually, and the fact that it did not contain any mobile part made this platform safer than the previous one (Dropbot project).

All the electronic components were completely isolated from the liquid phases, therefore it was impossible for them to contact each other.

As Figure 5.14 showed, the device, the electronics and the waste drum were placed in individual safety plates, thus if any of them leaked, it was impossible for the liquid to arrive to any of the others.

In our experience, the only possible failure that could happen was when the outlet hole of the device became blocked, and it could not drain its contents. The syringe pumps would continue to add liquid, but at the same time they would not be able to remove it, meaning that the arena would be filled and overflow. In order to fix this problem, we made sure that the outlet hole was big enough to never overflow.

The platform ran outside lab hours, but it was continuously checked remotely. It also had the "night experiment" information sheet attached to it.

The platform was always place inside a fume hood.

8.2.9 Protocol

Once the computer provided a recipe as described via a genetic algorithm, a lattice search or any other method, the 3D printed device was programmed to prepare the mixture, execute five experiments, and clean the arena in order to get ready for the next recipe.

In order to mix the oils through the serpentine channel, the oil with the highest value in the recipe was set to a volume of 400 µl, while all the other oils were given a volume proportional to 400 µl based on the ratio defined in the recipe. These volumes were then pumped into the device and through the serpentine channel at a flowrate derived from the proportional ratios. The oil with the highest volume was pumped in at 2.5 µl per second, while the other oils were pumped in at a slower speed proportional to the recipe. These oil mixtures were dispensed into the device arena, and once the mixture was completed the device removed the contents into a waste drum, and the arena was washed with 2.5 ml of acetone. This process was not only used to prepare a new mixture, but also to remove the contents from

the previous mixture. This mixing process was repeated three times. After this, a series of cleaning cycles were executed in order to be sure that the arena was fully cleaned. The first step was to wash it with 3 ml of acetone, and remove the contents. Then 2.5 ml of aqueous phase were added, and its content removed. Then 10 ml of acetone were added, and its content its content removed. This step was executed twice. Finally, 3.75 ml of aqueous phase were added, and its content removed. This step was executed twice.

The next objective was to perform the experiment. In order to do so, the arena was initially filled with 3.75 ml of aqueous phase, and then five oil injections of 10 µl were executed in order to generate the oil droplets. At this point, the experiment was performed and it was recorded using a camera. Each experiment lasted one minute.

The next objective was to clean the arena in order to perform the next experiment. In order to do so, the arena was initially filled with 7.5 ml of acetone, and its contents removed. Then it was filled with 3.5 ml of aqueous phase, and its contents removed. This last step was repeated twice.

Once this last step was performed, the device would fill the arena with aqueous phase to execute the same recipe again (each recipe was tested five times), or a new recipe would have been sent from the computer, in which case the device would start again with the mixing procedure.

8.2.10 Chemistry – preparation of solutions

The chemistry used in this project was exactly the same as used in the previous project (Chapter3, Section 8.1.4.1). We will only focus here on how the solutions were prepared for this project:

Initially NaOH (20.0 g, 10.0 mmol in 5 l) was dissolved in distilled water (ca 4.8 l), then TTAB (33.65 g, 20.0 mm in 5 l) was added. Then, the pH was adjusted to pH 13 using 6 M NaOH solution, and the volume was adjusted to 5 l. The pH-meter used was calibrated between pH 7 and pH 10. The oils 1-octanol, 1-pentanol and DEP were prepared in 200 ml aliquots in reagent bottles, while octanoic acid was diluted with 1-pentanol (20% octanoic acid 80% 1-pentanol), and also prepared in 100 ml aliquot in a reagent bottle. DEP and 1-octanol were dyed with 0.25 mg ml^{-1} Sudan II blue and vortexed to mix. 1-Pentanol and octanoic acid were dyed with 0.25 mg ml^{-1} Sudan III red and vortexed to mix.

8.2.11 Data analysis – Fitness landscape model[11]

The fitness landscapes shown in the results chapter (Section 6.4, p.150) were generated using support vector regression (SVR, [Drucker et al., 1997]). For each environment (empty, pillars and caves), data from 2 full GA runs were collated resulting in a dataset of 400 experiments each[12]. To find the most accurate representation, best parameters for the SVR were estimated using 10 fold cross-validation and with

[11]The data analysis as described here was done by Dr. Jonathan Grizou.

[12]Each experiment was repeated five times, and the average was returned. So although the model was based on this 400 experiments, in total it represented 2000 points.

respect to the mean squared error on the training set. Fitness landscapes are shown as ternary plot, a way to represent on a 2D plane three coupled variables which sum to a constant. In our case, we represented three out of the four components (with the fourth parameter held constant at zero). As our parameters represent percentage of each oils, the sum is thus constrained to 1. The C (penalty parameter of the error term) parameter was searched within [0.01, 0.1, 1, 10, 100]. For the empty environment, the best parameter were 'C': 100, 'gamma': 10 for an average mean square error of 6.83 (std=4.00). For the pillars environment, the best parameters were 'C': 10, 'gamma': 100 for an average mean square error of 21.30 (std=10.68). For the caves environment, the best parameters were 'C': 10, 'gamma': 100 for an average mean square error of 11.50 (std=4.50).

8.2.12 Video analysis implementation

A Microsoft LifeCam Cinema Web camera was situated above the arena in order to record the experiment. While the experiment happened, the camera stream was fed into a running Python (2.7.11) OpenCV (2.4.12) script which performed the image analysis and returned the number of droplets active at the end of the experiment. The video was configured to 800 x 600 pixels, and 30 frames per second (FPS). The first step consisted of defining a circular area with 275 pixels of radius. This area overlapped with the experimental circular arena, and only the pixels inside this area were considered for image processing. The image processing was performed using a Mixture of Gaussians (MoG) model for background subtraction. OpenCV's MoG[13] [Kaewtrakulpong and Bowden, 2001] was used for this purpose using the default configuration values. The MoG model was reinitialised before each experiment. It is important to remark that everything in the scene remained constant except the oil droplets, therefore all the pixels marked as foreground were droplets. The foreground subtracted was then used with a "find contours" operation from OpenCV in order to describe the droplets. At the end of the experiment, the number of droplets active was returned as a fitness value. By using a MoG with a small window, all the droplets that remained static for a few seconds were considered as part of the background and discarded. This way, only the droplets that always moved were considered part of the foreground. Figure 8.15 outlines the procedure, while Figure 8.16 shows the actual results of each step.

8.2.13 Middleware implementation

The *middleware* was implemented using Labview 2015, and the only non standard library used was the VISA NI one, which was used to send the G-Code commands from the computer running Labview to the Arduino board were the pumps were connected.

[13]When this code was written the most common version of OpenCV was OpenCV2. In Python it only had a MoG implementation. OpenCV3 comes with a theoretically better MoG implementation. If the results from this work were to be reproduced, OpenCV2 with MoG1 must be used, otherwise the results will be different. If on the other hand new data were to be collected for a different study, we would recommend OpenCV3 with MoG2.

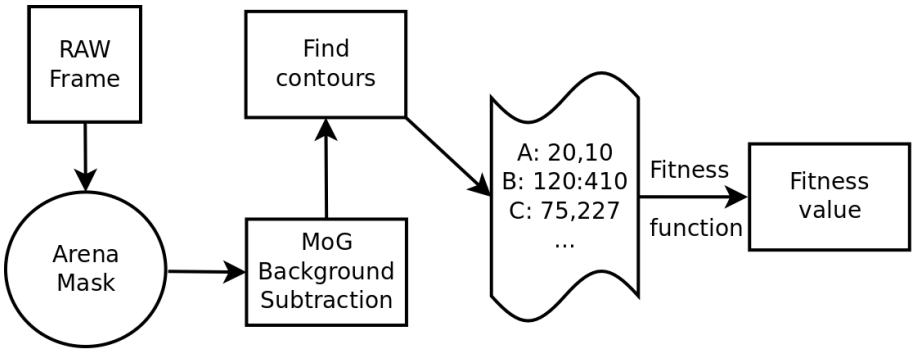


Figure 8.15: Flowbot image analysis pipeline. It started with a raw frame as received from the camera, which delivered them at 30 FPS, and their resolution was 800 by 600 pixels. The first step performed was a Mixture of Gaussians background subtraction, where all the objects that were not background were marked - with our set-up this could only be the oil phase. From these results, only the ones inside a circular arena were kept. This was done to remove possible noise from the outside, and to speed up the processing because only the pixels inside this area were computed. This information was then sent to the function "find contours" from OpenCV, which returned droplets' contours, from where we could extract their centre position. A list with all the droplet positions in every frame was finally returned, and based on this information a fitness value was produced.

As recommended by Labview in similar problems, the "Queued State Machine - Producer Consumer" architecture was used as design template ([Lukindo, 2007]). This architecture focuses on having a producer state machine, and a consumer one. Our producer took the recipes, transformed them into a sequence of "high level" actions, and then transformed these actions into pump actions. Our consumer took these pump actions and transformed them into G-code commands.

Figure 8.17 shows a snippet of the actual implementation identifying the different states of the producer state machine. This snippet does not aim to be a full VI representation. For more information, see the full VI files attached to the thesis. The first four states prepared the mixture and then cleaned the arena. Then the experiment was prepared, and finally it is cleaned again. Because each experiment was repeated five times, the state machine iterated between the states called "Aq phase" and "Clean Aq". It can also be seen in this figure how pump commands were executed. Because we were using a queued state machine, a command with the four parameters as discussed above was sent to the queue, which will be read later on during the consumer state machine.

Figure 8.18 shows a snippet of the actual implementation of the commands consumer state machine. As in the previous paragraph, this snippet does not aim to be a full VI representation. The snippet shows the two most interesting parts: (a) The states of the consumer state machine and (b) how the information that was loaded before in a queue data structure is now unfolded and injected into a string that represented a G-code command, and how this string is sent to the Arduino board using a VISA write operation. Four of the states of the consumer state machine directly represent the pump actions as explained before: "absorb liquid", "valve

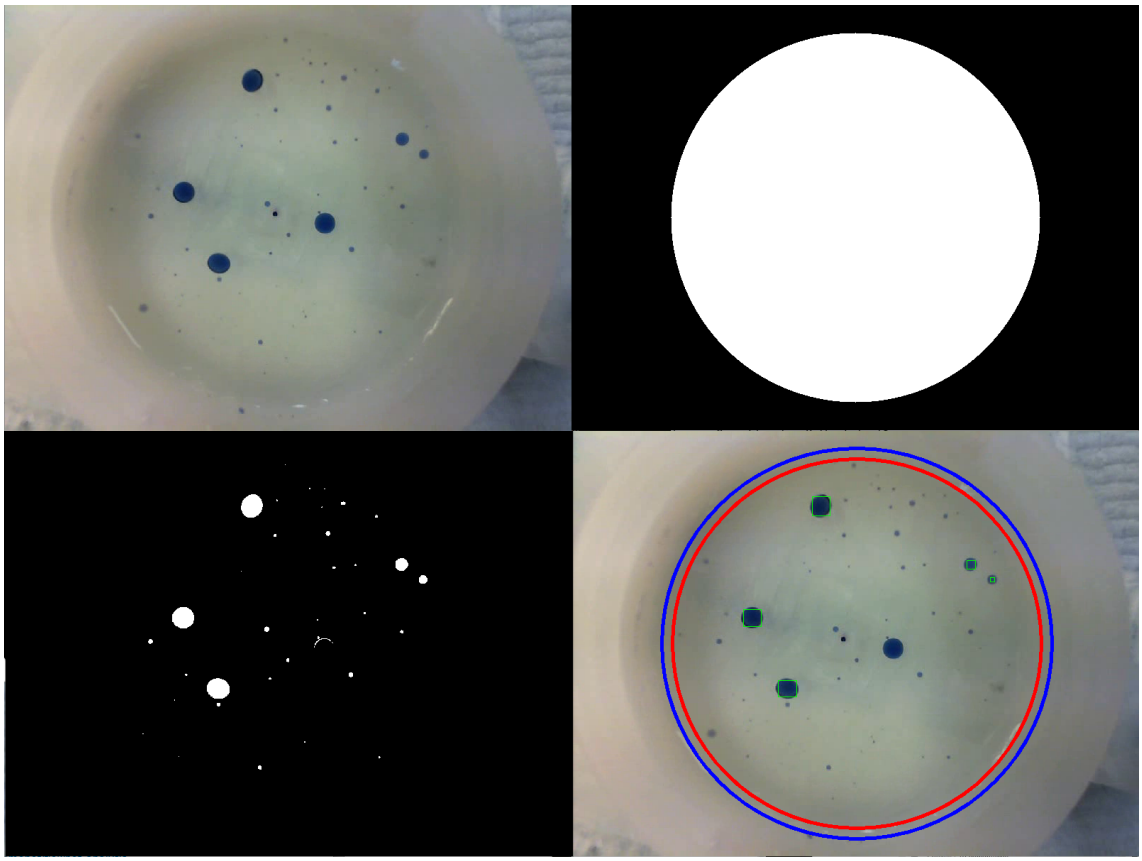


Figure 8.16: Flowbot image analysis. **Top left:** Frame as received from the camera. **Top right:** The mask used that defined the arena. All the black pixels were ignored. This mask was always the same through all the experiments. **Bottom left:** Results from the Mixture of Gaussians background subtraction. **Bottom right:** Final result with the detected droplets. All the droplets from "Bottom left" which were detected there but that don't appear now were disregarded because they were not big enough, or because they were not moving.

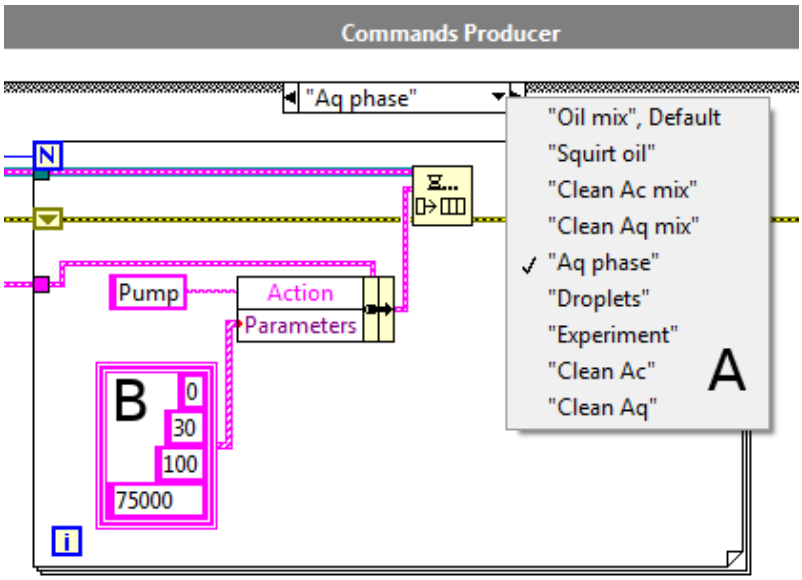


Figure 8.17: Labview implementation - Producer state machine. This figure does not aim to be a full VI representation. It only aims to show two things: **(A)** in the top-down menu there can be seen the states used to describe our experiments. They were executed in the sequence shown. **(B)** Example of pump action, were the required information as explained is sent to a queue data structure, which will be later on read by the consumer, and executed.

output", "Release liquid" and "valve input". The other two states are unrelated to the pumps themselves. The one called "wait" is the idle state were the loop stayed if there were no commands to consume. The "experiment" state is activated when a experiment command was sent from the producer state machine, and it called the Python library responsible for the image processing and waited until its execution finished.

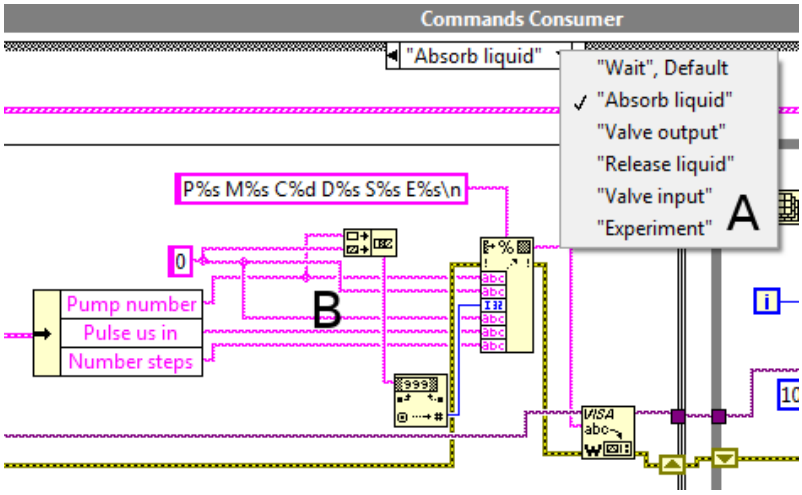


Figure 8.18: Flowbot Labview implementation - Consumer state machine. This figure does not aim to be a full VI representation. It only aims to show two things: **(A)** in the top-down menu there can be seen the states used to "consume" the commands generated before and produce G-code. Each of these states (except wait and experiment) were associated with a G-code command. **(B)** How the information produced before was read, inserted into a G-code command, then written into the VISA driver.

Bibliography

[Abouheif et al., 2014] Abouheif, E., Fave, M.-J., Ibarraran-Viniegra, A. S., Lesoway, M., Rafiqi, A. M., and Rajakumar, R. (2014). Eco-Evo-Devo: The time has come. In Landry, C. and Aubin-Horth, N., editors, *Ecological Genomics*, volume 781, pages 107–125. Springer.

[Adamala and Szostak, 2013] Adamala, K. and Szostak, J. W. (2013). Nonenzymatic template-directed RNA synthesis inside model protocells. *Science*, 342(6162):1098–1100.

[Adamatzky et al., 2012] Adamatzky, A., Holley, J., Dittrich, P., Gorecki, J., De Lacy Costello, B., Zauner, K. P., and Bull, L. (2012). On architectures of circuits implemented in simulated Belousov-Zhabotinsky droplets. *BioSystems*, 109(1):72–77.

[Altwegg et al., 2016] Altwegg, K., Balsiger, H., Bar-nun, A., Berthelier, J.-j., Bieler, A., Bochsler, P., Briois, C., Calmonte, U., Combi, M. R., Cottin, H., Keyser, J. D., Dhooghe, F., Fiethe, B., Fuselier, S. A., Gasc, S., Gombosi, T. I., Hansen, K. C., Haessig, M., Jäckel, A., Kopp, E., Korth, A., Roy, L. L., Mall, U., Marty, B., Mousis, O., Owen, T., Rème, H., Rubin, M., Sémon, T., Tzou, C.-y., Waite, J. H., and Wurz, P. (2016). Prebiotic chemicals — amino acid and phosphorus — in the coma of comet 67P / Churyumov-Gerasimenko. *Science Advances*, (May):1–6.

[Amasino, 2010] Amasino, R. (2010). Seasonal and developmental timing of flowering. *Plant Journal*, 61(6):1001–1013.

[Arduino, 2016] Arduino (2016). Servo library. *https://www.arduino.cc/en/Reference/Servo*.

[Arita et al., 2016] Arita, T., Joachimczak, M., Ito, T., Asakura, A., and Suzuki, R. (2016). ALife approach to eco-evo-devo using evolution of virtual creatures. *Artificial Life and Robotics*, 21(2):1–8.

[Auerbach and Bongard, 2014] Auerbach, J. E. and Bongard, J. C. (2014). Environmental influence on the evolution of morphological complexity in machines. *PLoS Computational Biology*, 10(1).

[Bachmann et al., 1990] Bachmann, P. A., Walde, P., Luisi, P. L., and Lang, J. (1990). Self-replicating reverse micelles and chemical autopoiesis. *Journal of the American Chemical Society*, 112(22):8200–8201.

[Baden et al., 2015] Baden, T., Chagas, A. M., Gage, G., Marzullo, T., Prieto-Godino, L. L., and Euler, T. (2015). Open labware: 3-D printing your own lab equipment. *PLOS Biology*, 13(3):e1002086.

[Balaban et al., 2004] Balaban, N. Q., Merrin, J., Chait, R., Kowalik, L., and Leibler, S. (2004). Bacterial Persistence as a Phenotypic Switch. *Science*, 305(September):1622–1626.

[Bangham et al., 1965] Bangham, A., Standish, M., and Watkins, J. (1965). Diffusion of univalent ions across the lamellae of swollen phospholipids. *Journal of Molecular Biology*, 13(1):238–IN27.

[Banno et al., 2013] Banno, T., Miura, S., Kuroha, R., and Toyota, T. (2013). Mode changes associated with oil droplets movement in solutions of gemini cationic surfactats. *Langmuir*, 29:7689–7696.

[Banno et al., 2016] Banno, T., Tanaka, Y., Asakura, K., and Toyota, T. (2016). Self-propelled oil droplets and their morphological change to giant vesicles induced by a surfactant solution at low pH. *Langmuir*.

[Banno and Toyota, 2015] Banno, T. and Toyota, T. (2015). Molecular system for the division of self-propelled oil droplets by component feeding. *Langmuir*, 31(25):6943–6947.

[Banzhaf et al., 2006] Banzhaf, W., Beslon, G., Christensen, S., Foster, J. a., Képès, F., Lefort, V., Miller, J. F., Radman, M., and Ramsden, J. J. (2006). From artificial evolution to computational evolution: a research agenda. *Nature reviews. Genetics*, 7(September):729–735.

[Barrick et al., 2009] Barrick, J. E., Yu, D. S., Yoon, S. H., Jeong, H., Oh, T. K., Schneider, D., Lenski, R. E., and Kim, J. F. (2009). Genome evolution and adaptation in a long-term experiment with Escherichia coli. *Nature*, 461(7268):1243–1247.

[Bedau et al., 1997] Bedau, M. a., Snyder, E., Brown, C. T., and Packard, N. H. (1997). A comparison of evolutionary activity in artificial evolving systems and in the biosphere. In *Proceedings of the Fourth International Conference on Artificial Life*, pages 125–134.

[Bell et al., 2015] Bell, E., Boehnke, P., Harrison, M., and Mao, W. (2015). Potentially biogenic carbon preserved in a 4.1 billion-year-old zircon. *Proceedings of the National Academy of Sciences*, 112(47):14518–14521.

[Bérut et al., 2012] Bérut, A., Arakelyan, A., Petrosyan, A., Ciliberto, S., Dillenschneider, R., and Lutz, E. (2012). Experimental verification of Landauer's principle linking information and thermodynamics. *Nature*, 483(7388):187–189.

[Bhargava et al., 2014] Bhargava, K. C., Thompson, B., and Malmstadt, N. (2014). Discrete elements for 3D microfluidics. *Proceedings of the National Academy of Sciences*, 111(42):15013–15018.

[Binh-Khiem et al., 2008] Binh-Khiem, N., Matsumoto, K., and Shimoyama, I. (2008). Polymer thin film deposited on liquid for varifocal encapsulated liquid lenses. *Applied Physics Letters*, 93(12).

[Bissette and Fletcher, 2015] Bissette, A. J. and Fletcher, S. P. (2015). Systems chemistry: Selecting complex behaviour. *Nature Chemistry*, 7:15–17.

[Bissette et al., 2014] Bissette, A. J., Odell, B., and Fletcher, S. P. (2014). Physical autocatalysis driven by a bond-forming thiol-ene reaction. *Nature communications*, 5:4607.

[Bradski and Kaehler, 2008] Bradski, G. and Kaehler, A. (2008). *Learning OpenCV*. O'Reilly Media.

[Bredeche et al., 2010] Bredeche, N., Montanier, J.-M., Liu, W., and Winfield, A. (2010). Environment-driven Embodied distributed evolutionary adaptation in a population of autonomous robotic agents. *Mathematical and Computer Modelling of Dynamical Systems*, 18:101–129.

[Brooks, 1986] Brooks, R. A. (1986). A Robust Layered Control System For A Mobile Robot. *IEEE Journal on Robotics and Automation*, 2(1):14–23.

[Brooks, 1991] Brooks, R. A. (1991). Intelligence without representation. *Artificial Intelligence*, 47(1-3):139–159.

[Browne et al., 2010] Browne, K., Walker, D., Bishop, K., and Grzybowski, B. (2010). Self-division of macroscopic droplets: partitioning of nanosized cargo into nanoscale micelles. *Angewandte Chemie (International ed. in English)*, 49(38):6756–9.

[Budin and Szostak, 2011] Budin, I. and Szostak, J. W. (2011). Physical effects underlying the transition from primitive to modern cell membranes. *Proceedings of the National Academy of Sciences*, 108(13):5249–54.

[Cafferty et al., 2013] Cafferty, B. J., Gallego, I., Chen, M. C., Farley, K. I., Eritja, R., and Hud, N. V. (2013). Efficient self-assembly in water of long noncovalent polymers by nucleobase analogues. *Journal of the American Chemical Society*, 135(7):2447–2450.

[Canny, 1986] Canny, J. (1986). A computational approach to edge detection. *IEEE transactions on pattern analysis and machine intelligence*, 8(6):679–698.

[Capel et al., 2013] Capel, A. J., Edmondson, S., Christie, S. D. R., Goodridge, R. D., Bibb, R. J., and Thurstans, M. (2013). Design and additive manufacture for flow chemistry. *Lab on a chip*, 13:4583–90.

[Casadevall i Solvas and DeMello, 2011] Casadevall i Solvas, X. and DeMello, A. J. (2011). Droplet microfluidics: recent developments and future applications. *Chemical communications*, 47(7):1936–42.

[Caschera et al., 2013] Caschera, F., Rasmussen, S., and Hanczyc, M. M. (2013). An oil droplet division-fusion cycle. *ChemPlusChem*, 78(1):52–54.

[Čejková et al., 2014] Čejková, J., Novák, M., Štěpánek, F., and Hanczyc, M. M. (2014). Dynamics of chemotactic droplets in salt concentration gradients. *Langmuir*, 30(40):11937–44.

[Čejková et al., 2016] Čejková, J., Štěpánek, F., and Hanczyc, M. M. (2016). Evaporation-induced pattern formation of decanol droplets. *Langmuir*, 32(19):4800–4805.

[Chen and Nowak, 2012] Chen, I. A. and Nowak, M. A. (2012). From prelife to life: how chemical kinetics become evolutionary dynamics. *Accounts of Chemical Research*, 45(12):2088–2096.

[Chen and Walde, 2010] Chen, I. A. and Walde, P. (2010). From Self-Assembled Vesicles to Protocells. *Cold Spring Harbor perspectives in biology*, 2(7):1–13.

[Cheney et al., 2015] Cheney, N., Bongard, J. C., and Lipson, H. (2015). Evolving Soft Robots in Tight Spaces. In *Proceedings of the 2015 on Genetic and Evolutionary Computation Conference2*, pages 935–942.

[Cheney et al., 2013] Cheney, N., MacCurdy, R., Clune, J., and Lipson, H. (2013). Unshackling evolution: evolving soft robots with multiple materials and a powerful generative encoding. *GECCO '13*, page 167.

[Cira et al., 2015] Cira, N. J., Benusiglio, A., and Prakash, M. (2015). Vapour-mediated sensing and motility in two-component droplets. *Nature*, 519(7544):446–450.

[Cobb, 1990] Cobb, H. G. (1990). An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environments. Technical report, Naval Research Laboratory, Washington DC.

[Comina et al., 2014] Comina, G., Suska, A., and Filippini, D. (2014). Low cost lab-on-a-chip prototyping with a consumer grade 3D printer. *Lab on a chip*, 14(16):2978–82.

[Cooper et al., 2011] Cooper, G. J. T., Kitson, P. J., Winter, R., Zagnoni, M., Long, D. L., and Cronin, L. (2011). Modular redox-active inorganic chemical cells: ICHELLs. *Angewandte Chemie - International Edition*, 50(44):10373–10376.

[Cronin et al., 2006] Cronin, L., Krasnogor, N., Davis, B. G., Alexander, C., Robertson, N., Steinke, J. H. G., Schroeder, S. L. M., Khlobystov, A. N., Cooper, G., Gardner, P. M., Siepmann, P., Whitaker, B. J., and Marsh, D. (2006). The imitation game–a computational chemical approach to recognizing life. *Nature biotechnology*, 24(10):1203–1206.

[Denkov et al., 2015] Denkov, N., Tcholakova, S., Lesov, I., Cholakova, D., and Smoukov, S. K. (2015). Self-shaping of oil droplets via the formation of intermediate rotator phases upon cooling. *Nature*, 528:392–395.

[Derényi and Lagzi, 2014] Derényi, I. and Lagzi, I. (2014). Fatty acid droplet self-division driven by a chemical reaction. *Physical chemistry chemical physics*, 16(10):4639–41.

[Dimitrov et al., 2006] Dimitrov, D., Schreve, K., and Beer, N. D. (2006). Advances in three dimensional printing - state of the art and future perspectives. *Rapid Prototyping Journal*, 12(3):136–147.

[Dorvee et al., 2004] Dorvee, J. R., Derfus, A. M., Bhatia, S. N., and Sailor, M. J. (2004). Manipulation of liquid droplets using amphiphilic, magnetic one-dimensional photonic crystal chaperones. *Nature materials*, 3(12):896–899.

[Dragone et al., 2013] Dragone, V., Sans, V., Rosnes, M. H., Kitson, P. J., and Cronin, L. (2013). 3D-printed devices for continuous-flow organic chemistry. *Beilstein Journal of Organic Chemistry*, 9:951–959.

[Drucker et al., 1997] Drucker, H., Burges, C. J., Kaufman, L., Smola, A., and Vapnik, V. (1997). Support vector regression machines. *Advances in neural information processing systems*, 9:155–161.

[Duda and Hart, 1972] Duda, R. and Hart, P. (1972). Use of the Hough Transformation to Detect Lines and Curves in Pictures. *Communication of ACM*, 15(1):11 – 15.

[Dworkin et al., 2001] Dworkin, J., Deamer, D., Sandford, S., and Allamandola, L. (2001). Self-assembling amphiphilic molecules: Synthesis in simulated interstellar/precometary ices. *Proceedings of the National Academy of Sciences*, 98(3):815–819.

[Eiben and Smith, 2015] Eiben, A. E. and Smith, J. (2015). From evolutionary computation to the evolution of things. *Nature*, 521(7553):476–482.

[Eppendorf, 2016] Eppendorf (2016). Eppendorf epMotion. *https://online-shop.eppendorf.co.uk/UK-en/Automated-Pipetting-44509.html*.

[Faina et al., 2016] Faina, A., Nejatimoharrami, F., Stoy, K., Theodosiou, P., Taylor, B., and Ieropoulos, I. (2016). EvoBot : An open-source , modular liquid handling robot for nurturing microbial fuel cells. In *Alife17*, pages 626–633.

[Fleischer et al., 2016] Fleischer, H., Drews, R. R., Janson, J., Chinna Patlolla, B. R., Chu, X., Klos, M., and Thurow, K. (2016). Application of a Dual-Arm Robot in Complex Sample Preparation and Measurement Processes. *Journal of laboratory automation*, 21(5):671–681.

[Floreano and Mattiussi, 2008] Floreano, D. and Mattiussi, C. (2008). Developmental systems. In *Bio-inspired artificial intelligence*, chapter 1. D. Flor, pages 269–335. The MIT Press, Cambridge.

[Ganti, 2002] Ganti, T. (2002). On the early evolutionary origin of biological periodicity. *Cell Biology International*, 26(8):729–735.

[Gardner et al., 2009] Gardner, P. M., Winzer, K., and Davis, B. G. (2009). Sugar synthesis in a protocellular model leads to a cell signalling response in bacteria. *Nature chemistry*, 1(5):377–383.

[Ghalambor et al., 2015] Ghalambor, C. K., Hoke, K. L., Ruell, E. W., Fischer, E. K., Reznick, D. N., and Hughes, K. (2015). Non-adaptive plasticity potentiates rapid adaptive evolution of gene expression in nature. *Nature*, 525:372–375.

[Glass et al., 2006] Glass, J. I., Assad-Garcia, N., Alperovich, N., Yooseph, S., Lewis, M. R., Maruf, M., Hutchison, C. a., Smith, H. O., and Venter, J. C. (2006). Essential genes of a minimal bacterium. *Proceedings of the National Academy of Sciences*, 103(2):425–430.

[Goul, 1982] Goul, S. J. (1982). Darwinism and the expansion of evolutionary theory. *Science*, 216(4544):380–387.

[Grasse et al., 2016] Grasse, E. K., Torcasio, M. H., and Smith, A. W. (2016). Teaching UV-Vis spectroscopy with a 3D-Printable smartphone spectrophotometer. *Journal of Chemical Education*, 93(1):146–151.

[Grefenstette, 1992] Grefenstette, J. (1992). Genetic algorithms for changing environments. *Parallel problem solving for nature*, 2:137–144.

[Hadorn et al., 2012] Hadorn, M., Boenzli, E., Sorensen, K. T., Fellermann, H., Eggenberger, P., and Hanczyc, M. M. (2012). Specific and reversible DNA-directed self-assembly of oil-in-water emulsion droplets. *Proceedings of the National Academy of Sciences*, 109(50):20320–20325.

[Hamilton, 2016a] Hamilton (2016a). Microlab NIMBUS. *http://www.hamiltoncompany.com/products/automated-liquid-handling/liquid-handling-workstations/microlab-nimbus*.

[Hamilton, 2016b] Hamilton (2016b). Microlab STAR Line. *http://www.hamiltoncompany.com/products/automated-liquid-handling/liquid-handling-workstations/microlab-star-line*.

[Hanczyc, 2008] Hanczyc, M. M. (2008). The early history of protocells: The search for the recipe of life. In *Protocells: Bridging nonliving and living matter*. MIT Press Scholarship.

[Hanczyc, 2014] Hanczyc, M. M. (2014). Droplets: Unconventional protocell model with life-like dynamics and room to grow. *Life*, 4:1038–1049.

[Hanczyc et al., 2008] Hanczyc, M. M., Chen, I. A., Sazani, P., and Szostak, J. W. (2008). Steps toward a synthetic protocell. In *Protocells: Bridging nonliving and living matter*. MIT Press Scholarship.

[Hanczyc and Monnard, 2016] Hanczyc, M. M. and Monnard, P.-A. (2016). The origin of life and the potential role of soaps. *Lipid Technology*, 28(5-6):88–92.

[Hanczyc et al., 2014] Hanczyc, M. M., Parrilla-Gutierrez, J. M., Nicholson, A., Yanev, K., and Stoy, K. (2014). Creating and Maintaining Chemical Artificial Life by Robotic Symbiosis. *Artificial Life*, 8:1–8.

[Hanczyc et al., 2007] Hanczyc, M. M., Toyota, T., Ikegami, T., Packard, N., and Sugawara, T. (2007). Fatty acid chemistry at the oil - water interface : self-propelled oil droplets. *Journal of the American Chemical Society*, 129(5):9386–9391.

[Harada et al., 2011] Harada, A., Kobayashi, R., Takashima, Y., Hashidzume, A., and Yamaguchi, H. (2011). Macroscopic self-assembly through molecular recognition. *Nature chemistry*, 3(1):34–37.

[Hargreaves et al., 1977] Hargreaves, W., Mulvihill, S., and Deamer, D. W. (1977). Synthesis of phospholipids and membranes in prebiotic conditions. *Nature*, 266:78–80.

[Haske et al., 2007] Haske, W., Chen, V. W., Hales, J. M., Dong, W., Barlow, S., Marder, S. R., and Perry, J. W. (2007). 65 nm feature sizes using visible wavelength 3-D multiphoton lithography. *Optics express*, 15(6):3426–3436.

[Henson et al., 2015] Henson, A., Parrilla-Gutierrez, J. M., Hinkley, T., Tsuda, S., and Cronin, L. (2015). Towards heterotic computing with droplets in a fully automated droplet-maker platform. *Philosophical Transactions of the Royal Society A*, 373(2046):1–10.

[Hiller and Lipson, 2012] Hiller, J. and Lipson, H. (2012). Automatic design and manufacture of soft robots. *IEEE Transactions on Robotics*, 28(2):457–466.

[Holland, 1975] Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press.

[Holmberg et al., 2002] Holmberg, K., Jonsson, B., Kronberg, B., and Lindman, B. (2002). *Surfactants and polymers in aqueous solution*. John Wiley & Sons.

[Horibe et al., 2011] Horibe, N., Hanczyc, M. M., and Ikegami, T. (2011). Mode switching and collective behavior in chemical oil droplets. *Entropy*, 13(3):709–719.

[Hornby et al., 2011] Hornby, G. S., Lohn, J. D., and Linden, D. S. (2011). Computer-automated evolution of an X-band antenna for NASA's Space Technology 5 mission. *Evolutionary computation*, 19(1):1–23.

[Hudson-Robotics, 2016a] Hudson-Robotics (2016a). Micro10x Microplate Dispenser. *http://hudsonrobotics.com/products/liquid-handling/micro10x/*.

[Hudson-Robotics, 2016b] Hudson-Robotics (2016b). SOLO Liquid handle. *http://hudsonrobotics.com/products/liquid-handling/solo/*.

[Ichihashi et al., 2013] Ichihashi, N., Usui, K., Kazuta, Y., Sunami, T., Matsuura, T., and Yomo, T. (2013). Darwinian evolution in a translation-coupled RNA replication system within a cell-like compartment. *Nature communications*, 4:2494.

[Ichimura et al., 2000] Ichimura, K., Oh, S.-K., and Nakagawa, M. (2000). Light-driven motion of liquids on a photoresponsive surface. *Science*, 288(5471):1624–1626.

[ISO, 1982] ISO (1982). ISO 6983 Numerical control of machines.

[Israelachvili et al., 1977] Israelachvili, J. N., Mitchell, D. J., and Ninham, B. W. (1977). Theory of self-assembly of lipid bilayers and vesicles. *Biochimica et Biophysica acta*, 470(2):185–201.

[Jiang et al., 2014] Jiang, L., Dziedzic, P., Spacil, Z., Zhao, G.-L., Nilsson, L., Ilag, L. L., and Córdova, A. (2014). Abiotic synthesis of amino acids and self-crystallization under prebiotic conditions. *Scientific reports*, 4:6769.

[Jin and Branke, 2005] Jin, Y. J. Y. and Branke, J. (2005). Evolutionary optimization in uncertain environments-a survey. *IEEE Transactions on Evolutionary Computation*, 9(3):303–317.

[Jonavic et al., 2016] Jonavic, T., Gailevic, D., Mizeikis, V., Gamaly, E. G., and Juodkazis, S. (2016). Nanoscale Precision of 3D Polymerization via Polarization Control. *Advanced Optical Materials*, pages 1–6.

[Jones et al., 2001] Jones, E., Oliphant, T., and Peterson, P. (2001). SciPy: Open source scientific tools for Python.

[Jones et al., 2011] Jones, R., Haufe, P., Sells, E., Iravani, P., Olliver, V., Palmer, C., and Bowyer, A. (2011). RepRap – the replicating rapid prototyper. *Robotica*, 29(01):177–191.

[Kaewtrakulpong and Bowden, 2001] Kaewtrakulpong, P. and Bowden, R. (2001). An improved adaptive background mixture model for real- time tracking with shadow detection. In *Video Based Surveillance Systems*, pages 135–144.

[Kashtan et al., 2007] Kashtan, N., Noor, E., and Alon, U. (2007). Varying environments can speed up evolution. *Proceedings of the National Academy of Sciences*, 104(34):13711–13716.

[Katsikis et al., 2015] Katsikis, G., Cybulski, J. S., and Prakash, M. (2015). Synchronous universal droplet logic and control. *Nature Physics*, 11:588–596.

[KhanAcademy, 2016] KhanAcademy (2016). What is life? *https://www.khanacademy.org/*.

[Kitson et al., 2016] Kitson, P. J., Glatzel, S., Chen, W., Lin, C.-G., Song, Y.-F., and Cronin, L. (2016). 3D printing of versatile reactionware for chemical synthesis. *Nature Protocols*, 11(5):920–936.

[Kitson et al., 2012] Kitson, P. J., Rosnes, M. H., Sans, V., Dragone, V., and Cronin, L. (2012). Configurable 3D-Printed millifluidic and microfluidic 'lab on a chip' reactionware devices. *Lab on a Chip*, 12(18):3267.

[Kohonen, 1982] Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43(1):59–69.

[Kong et al., 2012] Kong, F., Yuan, L., Zheng, Y. F., and Chen, W. (2012). Automatic liquid handling for life science: a critical review of the current state of the art. *Journal of laboratory automation*, 17(3):169–85.

[Kreutz et al., 2011] Kreutz, J. E., Shukhaev, A., Du, W., Druskin, S., Daugulis, O., and Ismagilov, R. F. (2011). Evolution of catalysts directed by genetic algorithms in a plug- based microfluidic device tested with oxidation of methane by oxygen. *Journal of the American Chemical Society*, 132(9):3128–3132.

[Krishna Kumar et al., 2011] Krishna Kumar, R., Yu, X., Patil, A. J., Li, M., and Mann, S. (2011). Cytoskeletal-like supramolecular assembly and nanoparticle-based motors in a model protocell. *Angewandte Chemie - International Edition*, 50(40):9343–9347.

[Kurihara et al., 2011] Kurihara, K., Tamura, M., Shohda, K.-i., Toyota, T., Suzuki, K., and Sugawara, T. (2011). Self-reproduction of supramolecular giant vesicles combined with the amplification of encapsulated DNA. *Nature Chemistry*, 3(10):775–781.

[Lagzi et al., 2010] Lagzi, I., Soh, S., Wesson, P. J., Browne, K. P., and Grzybowski, B. A. (2010). Maze solving by chemotactic droplets. *Journal of the American Chemical Society*, 132(4):1198–1199.

[Laland and O'Brien, 2010] Laland, K. N. and O'Brien, M. J. (2010). Niche construction theory and archaeology. *Journal of Archaeological Method and Theory*, 17(4):303–322.

[Lane and Martin, 2012] Lane, N. and Martin, W. F. (2012). The origin of membrane bioenergetics. *Cell*, 151(7):1406–1416.

[Langmuir, 1917] Langmuir, I. (1917). The constitution and fundamental properties of solids and liquids. Part II: Liquids. *Journal of the American Chemical Society*, 39(9):1848–1906.

[Lassabe et al., 2007] Lassabe, N., Luga, H., and Duthen, Y. (2007). A New Step for Evolving Creatures. In *IEEE Symposium on Artificial Life*, pages 243–251.

[Lau et al., 2004] Lau, M. W. L., Cadieux, K. E. C., and Unrau, P. J. (2004). Isolation of fast purine nucleotide synthase ribozymes. *Journal of the American Chemical Society*, 126(48):15686–93.

[Lee et al., 2015] Lee, M. P., Cooper, G. J. T., Hinkley, T., Gibson, G. M., Padgett, M. J., and Cronin, L. (2015). Development of a 3D printer using scanning projection stereolithography. *Scientific reports*, 5:9875.

[Lentini et al., 2014] Lentini, R., Santero, S. P., Chizzolini, F., Cecchi, D., Fontana, J., Marchioretto, M., Del Bianco, C., Terrell, J. L., Spencer, A. C., Martini, L., Forlin, M., Assfalg, M., Dalla Serra, M., Bentley, W. E., and Mansy, S. S. (2014). Integrating artificial with natural cells to translate chemical messages that direct E. coli behaviour. *Nature communications*, 5:4012.

[Li et al., 2011] Li, M., Green, D. C., Anderson, J. L. R., Binks, B. P., and Mann, S. (2011). In vitro gene expression and enzyme catalysis in bio-inorganic protocells. *Chemical Science*, 2(9):1739.

[Liechti, 2001] Liechti, C. (2001). pySerial. *https://github.com/pyserial/pyserial*.

[Lindenmayer, 1968] Lindenmayer, A. (1968). Mathematical models for cellular interactions in development. *Journal of theoretical biology*, 18(3):300–315.

[Lipson and Pollack, 2000] Lipson, H. and Pollack, J. B. (2000). Automatic design and manufacture of robotic lifeforms. *Nature*, 406(6799):974–978.

[Luisi, 2003] Luisi, P. L. (2003). Autopoiesis: a review and a reappraisal. *Die Naturwissenschaften*, 90(2):49–59.

[Lukindo, 2007] Lukindo, A. (2007). LabVIEW Queued State Machine Architecture. *https://decibel.ni.com/content/docs/DOC-32964*.

[Matthews and Minard, 2006] Matthews, C. N. and Minard, R. D. (2006). Hydrogen cyanide polymers, comets and the origin of life. *Faraday discussions*, 133(February 2006):393–401; discussion 427–452.

[McIntosh, 2013] McIntosh, C. (2013). "Life". In *Cambridge Advanced Learner's Dictionary*. 4th edition.

[Meyer, 1992] Meyer, F. (1992). Color image segmentation. In *International Conference on Image Processing and its Applications*, pages 303–306.

[Miller, 1953] Miller, S. L. (1953). A production of amino acids under possible primitive Earth conditions.

[Monnard and Deamer, 2002] Monnard, P. A. and Deamer, D. W. (2002). Membrane self-assembly processes: Steps toward the first cellular life. *Anatomical Record*, 268(3):196–207.

[Müller et al., 2015] Müller, A., Amaldass, A., Yanev, K., and Rasmussen, S. (2015). Do it yourself (DIY) liquid handling robot for evolutionary search exploration. *Proceedings of the European Conference on Artificial Life 15*, page 657.

[Nagai et al., 2005] Nagai, K., Sumino, Y., Kitahata, H., and Yoshikawa, K. (2005). Mode selection in the spontaneous motion of an alcohol droplet. *Physical Review E*, 71(6):065301.

[NEMA, 1998] NEMA (1998). NEMA Standards publication ICS 16: Motion/position control motors, controls, and feedback devices.

[Nie et al., 2012] Nie, B., Xing, S., Brandt, J. D., and Pan, T. (2012). Droplet-based interfacial capacitive sensing. *Lab on a Chip*, 12:1110–1118.

[Novellino et al., 2007] Novellino, A., D'Angelo, P., Cozzi, L., Chiappalone, M., Sanguineti, V., and Martinoia, S. (2007). Connecting neurons to a mobile robot: An in vitro bidirectional neural interface. *Computational Intelligence and Neuroscience*, 2007.

[Oka et al., 2014] Oka, M., Hashimoto, Y., and Ikegami, T. (2014). Self-organization on social media: Endo-exo bursts and baseline fluctuations. *PLoS ONE*, 9(10).

[OpenSCAD, 2016] OpenSCAD (2016). OpenSCAD - The Programmers Solid 3D CAD Modeller. *http://www.openscad.org/*.

[Opentrons, 2016] Opentrons (2016). Opentrons: robots for biologists. *https://opentrons.com/*.

[Otsu, 1979] Otsu, N. (1979). A Threshold Selection Method from Gray-Level Histograms. *IEEE Transactions on systems, man, and cybernetics*, 9(1):62–66.

[Parrilla-Gutiérrez, 2012] Parrilla-Gutiérrez, J. M. (2012). *Automatic Liquid Handling for Artificial Life*. PhD thesis, Syddanks Universitet.

[Parrilla-Gutierrez et al., 2014a] Parrilla-Gutierrez, J. M., Hinkley, T., Taylor, J., Yanev, K., and Cronin, L. (2014a). Hardware and Software manual for Evolution of Oil Droplets in a Chemo Robotic Platform. *arxiv.org*.

[Parrilla-Gutierrez et al., 2014b] Parrilla-Gutierrez, J. M., Hinkley, T., Taylor, J. W., Yanev, K., and Cronin, L. (2014b). Evolution of oil droplets in a chemorobotic platform. *Nature Communications*, 5:5571.

[Parter et al., 2008] Parter, M., Kashtan, N., and Alon, U. (2008). Facilitated variation: How evolution learns from past environments to generalize to new environments. *PLoS Computational Biology*, 4(11).

[Passy, 2002] Passy, S. (2002). Environmental Randomness Underlies Morphological Complexity of Colonial Diatoms. *Functional Ecology*, 16(5):690–695.

[Patel et al., 2015] Patel, B. H., Percivalle, C., Ritson, D. J., Duffy, C. D., and Sutherland, J. D. (2015). Common origins of RNA, protein and lipid precursors in a cyanosulfidic protometabolism. *Nature Chemistry*, 7(4):301–307.

[Peddi et al., 2007] Peddi, A., Muthusubramaniam, L., Zheng, Y. F., Cherezov, V., Misquitta, Y., and Caffrey, M. (2007). High-throughput automated system for crystallizing membrane proteins in lipidic mesophases. *IEEE Transactions on Automation Science and Engineering*, 4(2):129–139.

[PerkinElmer, 2016] PerkinElmer (2016). Zephyr NGS Workstation. *http://www.perkinelmer.com/product/zephyr-g3-ngs-workstation-133750*.

[Pfeifer et al., 2007] Pfeifer, R., Lungarella, M., and Iida, F. (2007). Self-organization, embodiment, and biologically inspired robotics. *Science*, 318(5853):1088–1093.

[Philamore et al., 2015] Philamore, H., Rossiter, J., Walters, P., Winfield, J., and Ieropoulos, I. (2015). Cast and 3D printed ion exchange membranes for monolithic microbial fuel cell fabrication. *Journal of Power Sources*, 289:91–99.

[Porter et al., 2016] Porter, L. A., Washer, B. M., Hakim, M. H., and Dallinger, R. F. (2016). User-friendly 3D printed colorimeter models for student exploration of instrument design and performance. *Journal of Chemical Education*, 93:1305–1309.

[Post and Palkovacs, 2009] Post, D. M. and Palkovacs, E. P. (2009). Eco-evolutionary feedbacks in community and ecosystem ecology: interactions between the ecological theatre and the evolutionary play. *Philosophical transactions of the Royal Society B*, 364(1523):1629–40.

[Powner et al., 2009] Powner, M. W., Gerland, B., and Sutherland, J. D. (2009). Synthesis of activated pyrimidine ribonucleotides in prebiotically plausible conditions. *Nature*, 459(7244):239–242.

[Pross, 2009] Pross, A. (2009). Seeking the chemical roots of Darwinism: Bridging between chemistry and biology. *Chemistry - A European Journal*, 15(34):8374–8381.

[Python-Turtle, 2016] Python-Turtle (2016). turtle - Turtle graphics for Tk. *https://docs.python.org/2/library/turtle.html*.

[Ray, 1991] Ray, T. (1991). An Approach to the synthesis of the. In *Artificial Life II*, pages 371–408.

[RepRap, 2016] RepRap (2016). Thermistor - RepRapWiki. *http://reprap.org/wiki/Thermistor*.

[Rodriguez-Garcia et al., 2015] Rodriguez-Garcia, M., Surman, A. J., Cooper, G. J., Suárez-Marina, I., Hosni, Z., Lee, M. P., and Cronin, L. (2015). Formation of oligopeptides in high yield under simple programmable conditions. *Nature Communications*, 6.

[Ruiz-Mirazo et al., 2014] Ruiz-Mirazo, K., Briones, C., and de la Escosura, A. (2014). Prebiotic Systems Chemistry: New Perspectives for the Orgins of Life. *Chemical Reviews*, 114(1):285–366.

[Scharf and Cronin, 2016] Scharf, C. and Cronin, L. (2016). Quantifying the Origins of Life on a Planetary Scale. *Proceedings of the National Academy of Sciences*, 113(29):8127–8132.

[Schopf, 2006] Schopf, J. W. (2006). Fossil evidence of Archaean life. *Philosophical transactions of the Royal Society B*, 361(1470):869–885.

[Scriven and Sternling, 1960] Scriven, L. E. and Sternling, C. V. (1960). The Marangoni Effects. *Nature*, 187:186–188.

[Seah et al., 2013] Seah, T. H., Zhao, G., and Pumera, M. (2013). Surfactant capsules propel interfacial oil droplets: An environmental cleanup strategy. *ChemPlusChem*, 78(5):384.

[Segré et al., 2001] Segré, D., Ben-Eli, D., Deamer, D. W., and Lancet, D. (2001). The Lipid World. *Origins of Life and Evolution of the Biosphere*, 31(1-2):119–145.

[Segré et al., 2000] Segré, D., Ben-Eli, D., and Lancet, D. (2000). Compositional genomes: prebiotic information transfer in mutually catalytic noncovalent assemblies. *Proceedings of the National Academy of Sciences*, 97(8):4112–7.

[Shapiro, 2007] Shapiro, R. (2007). A Simpler Origin for Life: Scientific American. *Scientific American*, 296(6):46–53.

[Shirt-Ediss et al., 2015] Shirt-Ediss, B., Solé, R., and Ruiz-Mirazo, K. (2015). Emergent Chemical Behavior in Variable-Volume Protocells. *Life*, 5(1):181–211.

[Shirt-Ediss, 2016] Shirt-Ediss, B. J. (2016). *Modelling Early Transitions Toward Autonomous Protocells*. PhD thesis, Universidad del Pais Vasco and Universitat Pompeu Fabra.

[Sims, 1994a] Sims, K. (1994a). Evolving 3D Morphology and Behavior by Competition. *Artificial Life*, 1(4):353–372.

[Sims, 1994b] Sims, K. (1994b). Evolving virtual creatures. In *Siggraph '94*, number July, pages 15–22.

[Song et al., 2014] Song, C., Moon, J. K., Lee, K., Kim, K., and Pak, H. K. (2014). Breathing, crawling, budding, and splitting of a liquid droplet under laser heating. *Soft Matter*, 10:2679–2684.

[Style et al., 2013] Style, R., Che, Y., Park, S. J., Weon, B. M., Je, J. H., Hyland, C., German, G., Power, M., Wilen, L., Wettlaufer, J., and Dufresne, E. (2013). Patterning droplets with durotaxis. *Proceedings of the National Academy of Sciences*, 110(31):12541–4.

[Suematsu et al., 2016] Suematsu, N. J., Mori, Y., Amemiya, T., and Nakata, S. (2016). Oscillation of speed of a self-propelled Belousov–Zhabotinsky droplet. *The Journal of Physical Chemistry Letters*, 7:3424–3428.

[Sumino et al., 2009] Sumino, Y., Kitahata, H., Seto, H., Nakata, S., and Yoshikawa, K. (2009). Spontaneous deformation of an oil droplet induced by the cooperative transport of cationic and anionic surfactants through the interface. *Journal of Physical Chemistry B*, 113(48):15709–15714.

[Suzuki et al., 2012] Suzuki, A., Maeda, S., Hara, Y., and Hashimoto, S. (2012). Capsule gel robot driven by self-propelled oil droplet. *IEEE International Conference on Intelligent Robots and Systems*, 7523:2180–2185.

[Symes et al., 2012] Symes, M. D., Kitson, P. J., Yan, J., Richmond, C. J., Cooper, G. J. T., Bowman, R. W., Vilbrandt, T., and Cronin, L. (2012). Integrated 3D-printed reactionware for chemical synthesis and analysis. *Nature Chemistry*, 4(5):349–354.

[Szostak, 2016] Szostak, J. (2016). Environmentally driven growth. *http://molbio.mgh.harvard.edu/szostakweb/researchVesicles.html*.

[Szostak et al., 2001] Szostak, J., Bartel, D., and Luisi, P. L. (2001). Synthesizing life. *Nature*, 409:387–390.

[Szymanski et al., 2013] Szymanski, J., Gorecki, J., and Hauser, M. (2013). Chemomechanical coupling in reactive droplets. *Journal of Physical Chemistry C*, 117(25):13080–13086.

[Taylor, 2015] Taylor, T. (2015). Requirements for Open-Ended Evolution in Natural and Artificial Systems. *EvoEvo Workshop at the European Conference on Artificial Life 2015 (ECAL 2015)*.

[Tecan, 2016] Tecan (2016). Tecan Freedom Evo 7.

[Toyota et al., 2009] Toyota, T., Maru, N., Hanczyc, M. M., Ikegami, T., and Sugawara, T. (2009). Self-propelled oil droplets consuming "fuel" surfactant. *Journal of the American Chemical Society*, 131(14):5012–3.

[Trifonov, 2011] Trifonov, E. N. (2011). Vocabulary of definitions of life suggests a definition. *Journal of biomolecular structure & dynamics*, 29(2):259–66.

[Tsuda et al., 2015] Tsuda, S., Jaffery, H., Doran, D., Hezwani, M., Robbins, P. J., Yoshida, M., and Cronin, L. (2015). Customizable 3D printed 'plug and play' millifluidic devices for programmable fluidics. *Plos One*, 10(11):e0141640.

[Tsuda et al., 2007] Tsuda, S., Zauner, K.-P., and Gunji, Y.-P. (2007). Robot control with biological cells. *Bio Systems*, 87(2-3):215–23.

[Villar et al., 2013] Villar, G., Graham, A., and Bayley, H. (2013). A Tissue-Like Printed Material. *Science*, 340(6128):48–52.

[Walter and Engelke, 2002] Walter, N. G. and Engelke, D. R. (2002). Ribozymes: Catalytic RNAs that cut things, make things, and do odd and useful jobs. *Biologist*, 49(5):199–203.

[Wijnen et al., 2014] Wijnen, B., Hunt, E. J., Anzalone, G. C., and Pearce, J. M. (2014). Open-source syringe pump library. *PLoS ONE*, 9(9):1–8.

[Xue and Leibler, 2016] Xue, B. and Leibler, S. (2016). Evolutionary learning of adaptation to varying environments through a transgenerational feedback. *Proceedings of the National Academy of Sciences*, (Early edition).

[Yanev, 2016a] Yanev, K. (2016a). Printrun. *https://github.com/kliment/Printrun*.

[Yanev, 2016b] Yanev, K. (2016b). Sprinter. *https://github.com/kliment/Sprinter*.

[Yuen and Kvenvolden, 1973] Yuen, G. and Kvenvolden, K. (1973). Monocarboxylic Acids in Murray and Murchison Carbonaceous Meteorites. *Nature*, 246(5431):301–303.

[Yuen, 2016] Yuen, P. K. (2016). A reconfigurable stick-n-play modular microfluidic system using magnetic interconnects. *Lab on a Chip*, adv. artic.

[Zhu et al., 2009] Zhu, T. F., Szostak, J. W., Zhu, T. F., and Szostak, J. W. (2009). Coupled Growth and Division of Model Protocell Membranes. *Journal of the American Chemical Society*, 131(15):5705–5713.