Autonomous Control of a Free-Flying Space Robot

Teri Welsh

A thesis submitted in complete fulfilment for the degree of Doctor of Philosophy to the Department of Aerospace Engineering, Faculty of Engineering, University of Glasgow.

June 2006

Abstract

The ongoing requirement for the assembly of large space structures has made a call for astronauts to work in partnership with a new generation of free-flying robotic vehicles. This thesis develops the control methodology for a flying robot designed to operate autonomously onboard crewed spacecraft in pressurized or vacuum micro-gravity environments. The controller will provide the robot with decision-making capabilities, allowing it to navigate autonomously within the vicinity of a large space structure and complete a number of tasks. The controller design uses a behavioural 'Braitenberg' approach to avoid collisions and achieve useful task objectives such as reaching goal destinations, collecting randomly positioned objects, refuelling and following moving targets. The incorporation of manual input is developed to allow external control over the automated robotic vehicle.

The suite of behaviours are given a variable weighting, to provide a versatile control methodology with seamless transition between behaviours, and in addition, integration of cue-deficit techniques to optimise the behavioural control when confronted with conflicting choices - such as the need to refuel whilst searching out a goal.

The model is enhanced by the addition of a camera tool to complement the third person viewpoint with the ability to point the robot's camera optical axis in any desired orientation, providing tracking and fixed-pointing capabilities with possible uses in video conferencing. The camera tool incorporates an attitude controller (using potential functions) to bring the robot to rest at the desired goal orientation, or track moving targets.

In summary, this thesis documents the development of a novel control methodology which integrates high-level behaviour based autonomy with low level translation and rotational control.

Acknowledgements

I would like to thank my supervisor, Professor Colin McInnes for his advice, guidance and expertise and for his belief and confidence in my achievements.

I would also like to thank the following organisations for their help and assistance in the course of my research, without which, much of the work herein would not be possible: The Caledonian Research Foundation, Professors Bruce Lusignan and Bob Twiggs and the Space Systems Development Laboratory team at the University of Stanford California, Yuri Gawdiak and the NASA Ames Research Center and finally, the Department of Aerospace Engineering at the University of Glasgow.

I would like to thank Dr Roan Lavery for his friendship, encouragement and intellect and for being a true inspiration for my life achievements.

And of course, my friends who have endured tormented regular updates on the latest techniques employed herein. And, finally, to David McMenemy, for his motivation, enthusiasm and cheerfulness, and for living life in bright colours.

I would like to dedicate this thesis to my family, whose relentless support, encouragement and kindness has prevailed throughout my life and allowed me to achieve my ambitions.

meus profundus gratiae.

Contents

List of Figures

Nomenclature

$\underline{a}$	Collision avoidance acceleration
$\underline{a}_{av}$	Required collision avoidance acceleration
$\underline{a}_t$	Threshold acceleration
C_{signal}	Chemical concentration
c	Braitenberg coefficient
D	Binary molecular diffusion coefficient
d	Distance
f	External force
fovy	Field of view along y-axis
G	Universal Gravitational Constant ($6.673 \times 10^{-11} kg^{-2}s$)
g_r	Acceleration due to gravity acting on the robot
g_s	Acceleration due to gravity acting on the space station
$\underline{H}$	Angular momentum
h	Hunger
I	Moment of Inertia
Isp	Propellant specific impulse
k	Accessibility
m_1	Earth Mass ($5.976 \times 10^{24} kg$)
m_2	Mass of space station
m_3	Robot mass
M	Matrix representation of behaviour
$\underline{n}$	Unit normal
P	Current physiological state
Q	Calibration Constant (100)
q	Thrust level
R	Sensor horizon; Energy
$\underline{R}$	Rotation Matrix
r	Availability
$\underline{r}_r$	Robot position vector
$\underline{r}_s$	Radius of orbit of space station around Earth
$\underline{r}_t$	Target position vector

$\underline{r}_1$	Position vector of Earth
$\underline{r}_2$	Position vector of Space Station
S	Signal strength
T	Trajectory
t	Thirst, time
$\underline{u}$	Final velocity
$\underline{v}$	Velocity
Δv	Translational velocity
W	Work
λ	Weighting applied to behaviour
μ	Gravitational Parameter
ω	Angular velocity
θ	Euler angles
θ^*	Desired orientation

Acronyms and Abbreviations

AI	Artificial Intelligence
AL	Artificial Life
CCTV	closed circuit television
ERA	European Robotic Arm
ESA	European Space Agency
EVA	Extra Vehicular Activity
GLUT	OpenGl Utility Toolkit
ISS	International Space Station
MCS	Monitoring and Control Station
PDA	Personal Digital Assistant
PSA	Personal Satellite Assistant
NASA	National Aeronautics and Space Administration
SPDM	Special Purpose Dexterous Manipulator
SSRMS	Space Station Remote Manipulator System
TLC	Transport and Launch Container

Chapter One: Introduction

'I suspect that if some of the bee and spider people
were to join forces with some of the AI people,
it would be a mutually enriching experience'

Daniel Dennet

1.1 Introduction

The most ambitious technical project in international history is currently underway with the construction and operation of the International Space Station (ISS).

The construction process being carried out in a joint venture by several leading space organisations to station an international research laboratory in orbit, will, during its twelve years as an operational laboratory facility, continue the human presence in space previously maintained by the Shuttle-Mir programme (*Space Station Assembly*, 2006), (*International Space Station*, 2006). With the continuing development of the ISS into a large research facility, astronauts are required to work in partnership with a new generation of space robots. Free-flying external observation and inspection vehicles are aiding astronauts and the existing "space crane" mechanical arms with the latter stages of the ISS configuration assembly for documentation and observation purposes. And indeed, during its lifetime, there will be an ongoing requirement for assistance in inspection, maintenance and repair operations (Roger and M^{c}Innes, 1999).

In addition to external functions, there are calls for internal robotic astronaut assistance onboard the space facility. Free-flying robots, such as NASA's Personal Satellite Assistant, (Bradshaw, Sierhuis & Gawdiak, 2000), (*Personal Satellite Assistant*, 2002)(Bluethmann, W., Ambrose, R., Diftler, M., Askew, S., Huber, E.,

Goza, M., Rehnmark, F., Lovchik, C., & Magruder, 2003) are required for environmental monitoring, communication links, remote operations support and crew worksite support, performing a vital role as an integrated part of the onboard operations crew. These requirements will become even more accurate for future, crewed deep space missions. (Dorias & Nicewarner, 2003)

This thesis will develop the control methodology for a flying robot designed to operate autonomously onboard crewed spacecraft in pressurized, or vacuum, micro-gravity environments. The controller will provide the robot with decision-making capabilities, allowing it to navigate autonomously within the vicinity of a large space structure and complete a number of tasks. The controller design uses a behavioural approach discussed in **chapters 3 & 4** to avoid collisions and achieve task objectives such as following wall surfaces, reaching goal destinations, (**chapter 6**) collecting randomly positioned objects, (**chapter 8**) refuelling, (**chapter 7**) and following moving targets (**chapter 6**). The incorporation of manual input is developed in **chapter 8**, by using a joystick to allow external control over the automated robotic vehicle.

These behaviours are each given a variable weighting, influenced by the vehicle state, to provide a versatile control methodology with seamless transition between behaviours. **Chapter 7** integrates cue-deficit techniques to optimise the behavioural control when confronted with conflicting choices - such as the need to refuel, determined by monitored fuel levels, whilst searching out a goal destination.

The algorithms are coded in C, and tested in a virtual environment developed in **chapter 2** to resemble an interior module of the International Space Station complete with potential collision hazards, a refuelling depot and a goal base, developed using the OpenGl® software interface.

In **chapter 5** the model is enhanced by the addition of a virtual camera tool to complement the third person viewpoint with the ability to point the robot's camera optical axis in any desired orientation, providing tracking and fixed-pointing capabilities with possible uses in video conferencing. The camera tool incorporates an

attitude controller (using potential functions) to bring the robot to rest at the desired goal orientation or tracking moving targets. The orbital dynamics influencing the motion of the robot in space are also incorporated to realistically simulate the robot's trajectory.

The controller uses little computational or processing power, made possible by basing its decision-making algorithms on a perception/action mechanism similar to those used by simple biological organisms. Adopting this mechanism of Artificial Life and using simple sensors and beacons, abandons the need for precise, complex world models, detailed maps and intensive calculations of inverse kinematics required in classical Artificial Intelligence (AI) methods. This approach also greatly reduces the need for computational resources and high processing power, whilst allowing the robot to navigate in an unknown or changing environment. Final conclusions are drawn in **chapter 9**.

1.2 The International Space Station

The beginning of a permanent presence in space was not established until 1971 with the commencement of the Soviet *Salyut* programme, and then in 1973 by the launch of the American *Skylab* station. The *Skylab* programme was brief, continuing only until 1974, and was surpassed by the Soviet's series of 5 *Salyuts*, continuing throughout the 70's through the adaptation of their Moon-era hardware into orbital space stations (Space Station History, 2006). In 1986 Salyut was succeeded by the modular space station, Mir, as Russia began to focus on long-duration space missions, providing the first space station to be continually crewed since launch. The cash strapped Mir project became obsolete in 1998, and there followed a series of proposals between NASA and the Russian Space Agency to construct a new space station. Through an international search for financial backing, this project then grew into a joint venture between 16 nations to construct and operate the first international orbital research laboratory, the largest space program ever realised.

In 1998, Astronaut Nancy Currie docked the first two modules of the International Space Station together in orbit. The first crew arrived in 2000 following the launch of many more modules and supply devices. The modules and supply devices will be provided by NASA, ESA, the Japanese NASDA and the Russian RKA. In total, 16 nations have participated providing almost 500 tonnes of material and components transported to the orbiting construction site on Russian and American launch vehicles (see **figure 1.1**).

A crew of astronauts are working continuously on the ISS staying for several months at a time, performing numerous spacewalks to assemble and maintain parts of the station. They are assisted by numerous robotic tools such as a robotic 'arm', a two-fingered robot 'hand' and a free-flying robotic "eye" which circle and inspect the station, in addition to the internal robotic assistance onboard the space facility required for environmental monitoring, crew communications and operations support (*Space station EVA*, 2006), (*ISS Overview*, 2006).

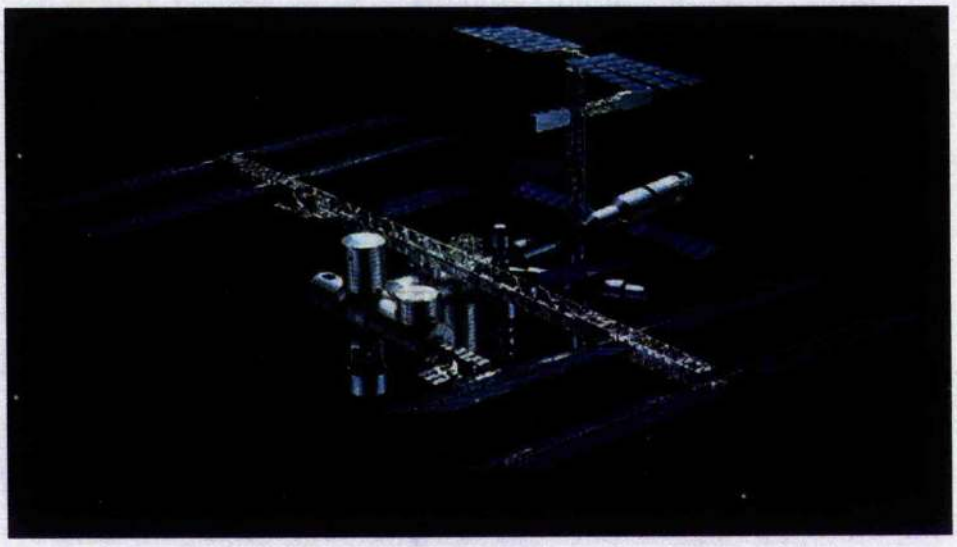


Figure 1.1: Artist's impression of the International Space Station.

1.3 Space Robotics

1.3.1 Future Requirements

The ongoing assembly of the International Space Station has made the call for robotic vehicles to perform remote observations and external operations in space an imperative requirement. In addition, the use of robotic servicing vehicles throughout the station's lifetime will become essential, in order to significantly reduce astronaut EVA hours.

Current work in this area includes the Space Station's mechanical arm which operates as a space crane, accurately manoeuvring large payloads into place, and posting astronauts to specific work areas (*A New Generation of Space Robots*, 2006). There is also additional experimental prototyping of free-flying camera tools to be used for remote inspection of the Space Shuttle and Space Station, such as the AERCam project (Choset & Kortenkamp, 1999), (*AERCam*, 2006), and the DASA Inspector Vehicle (Wilde & Sytin, 1995), (*Inspector*, 2006) and future requirements for internal robotic astronaut assistance onboard the space facility such as the prototype Personal Satellite Assistant currently being developed at NASA Ames, California (Bradshaw, Sierhuis & Gawdiak, 2001), (*Personal Satellite Assistant*, 2006).

1.3.2 Robotic Arms

The Space Shuttle and Station currently have two functioning mechanical, arms working in partnership with the Station crew providing strong, precise and delicate handling of the shuttle payloads.

The Canadian built Shuttle arm has proven reliable and versatile during a number of shuttle missions over the last 20 years, placing and plucking satellites from orbit since its maiden voyage aboard U.S. Space Shuttle Columbia in 1981. It

possesses an advanced 'Space Vision System' with the ability to track payloads and enable the operator to perform precision control of the robotic arm even when vision is partially obscured. Astronaut Nancy Currie initiated the assembly of the ISS, mating the U.S. Utility Node to the Russian built Zarya with the aid of the Space Vision System (*Canadarm*, 2006).

The vision system uses targets on the module surfaces, together with video imaging, to aid the operator by providing a virtual display of the desired surface. **Figure 1.2** shows the Vision System targets on the module Destiny, displayed as black and white circles on the module surface, allowing the operator precision control of the robotic arm (*A New Generation of Space Robots*, 2006).

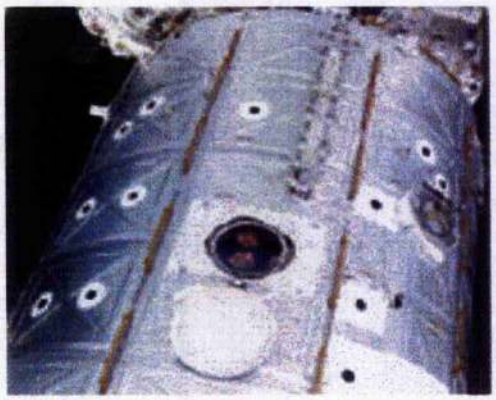


Figure 1.2: Module Destiny complete with Space Vision System targets.

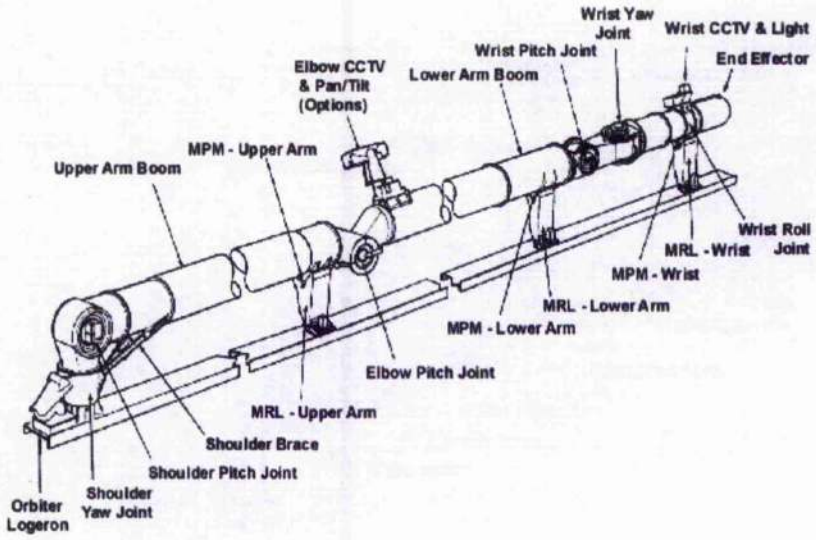


Figure 1.3(a): Canadarm2, showing the shoulder, elbow and wrist joints.

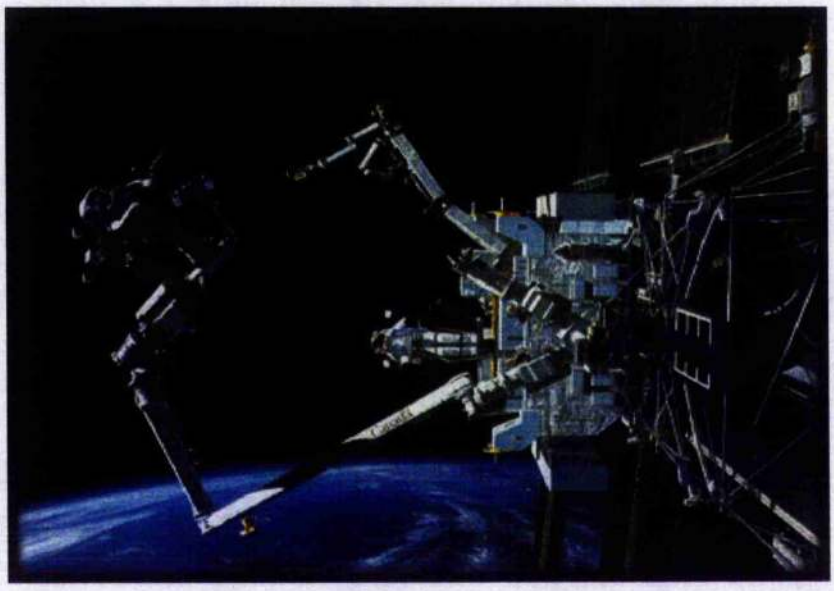


Figure 1.3(b): Canadarm2, showing the shoulder, elbow and wrist joints.

On STS-100, Canadarm assisted with the delivery and installation of a second-generation arm, the Space Station Remote Manipulator System (SSRMS), capable of manoeuvring payloads of up to 14,500 kg and used in the deployment and retrieval of space hardware. It will play a key role in space station assembly and maintenance, and astronaut support (Patten, 2003).

The SSRMS is a re-locatable, remotely controlled, payload handling device with six degrees-of-freedom and comprising of 'shoulder', 'elbow' and 'wrist' joints separated by upper and lower arm booms, as shown in **figure 1.3(a) & (b)**. The device uses more advanced technology, is capable of carrying much greater payloads, and moves with higher precision and flexibility than Canadarm1. It has a computerised control system, or may alternatively be positioned manually by astronauts, for payload capture or deployment, with the use of hand controllers and two closed circuit televisions (CCTV) located at the elbow and wrist joints. The SSRMS is designed to have a ten-year operational life.

It is capable of moving around the exterior surfaces of the ISS by latching its specialised end effectors onto complementary ports found on the surface of the Space Station, and by detaching one effector, it can then pivot over its anchoring point and attach to a port further forward. Repeating this procedure allows the arm to transverse the Station, with the ports providing the arm with power and video links to the astronaut operator inside the station (*Canadarm2*, 2006), (*The Shuttle Remote Manipulator System*, 2006).

Canadarm2 forms one of three parts of the Mobile Servicing System, which also consists of a mobile work platform used to escort Canadarm2 along tracks traversing the ISS exterior surface, and a Special Purpose Dexterous Manipulator, (SPDM), a robotic "hand" that can manipulate delicate objects (shown in **figure 1.4**).

The SPDM is a two-digit design attached to the end of the arm and is used for station maintenance and payload servicing, replacing smaller components on the

Station's surface where delicacy and flexibility is required. It is equipped with four TV cameras, to aid astronaut's control from inside the station, hence saving astronaut EVA in hostile space environments (*SPDM*, 2006).

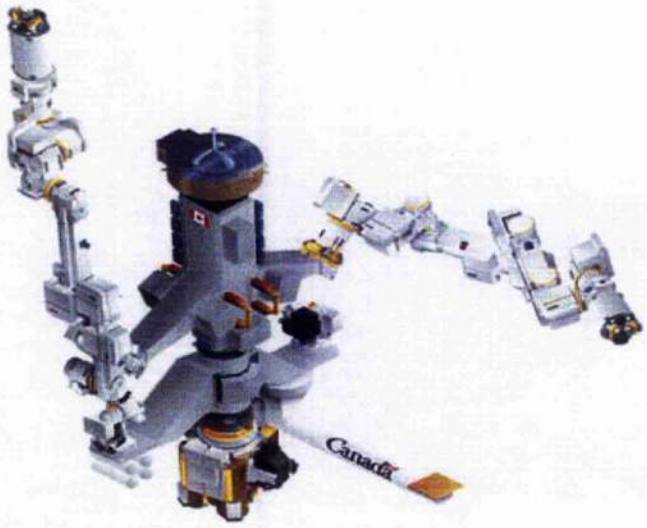


Figure 1.4: The Canadian Special Purpose Dexterous Manipulator.

The European partners of the International Space Station Project are developing a European Robotic Arm (ERA), for maintenance of the Russian segment of the Space Station including assembly, inspection and replacement of the Russian solar arrays as well as transferring cosmonauts for EVA activities, and other maintenance tasks. This is a joint co-operative venture between ESA and the Russian Space Agency with Fokker Space as their prime contractor. The 11 m long robotic tool, with mission handling capabilities of 8000 kg, has seven joints and two end effectors that will allow it to crawl along the ISS surface in a similar fashion to Canadarm2.

These large robots, the Canadian Space Station Remote Manipulator System and the European Robotics Arm, have limited viewing and reach capabilities. However, small free-flying robots can complement these larger systems by offering

high flexibility and access to difficult reaching areas. The first tasks envisaged by these small and autonomous systems are surveillance and routine inspection, with later versions having maintenance capabilities, fitted with dexterous robotic arms and fingers (Roderick, Roberts, Atkins, Churchill & Akin, 2004).

1.3.3 Aercam

Research in remote inspection robots specifically designed for space operations is a fairly new concept, which is attracting worldwide attention. Designed for remote surface inspection of inaccessible or dangerous to reach areas, these vehicles can reduce astronaut EVA hours significantly and limit the number of space walks required. Space agencies are eager to promote this area of robotics through projects such as NASA's Autonomous EVA Robotic Camera series *Aercam* (Choset & Kortenkamp, 1999), (*Aercam*, 2006) as shown in **figure 1.5**.

The first prototype in the Aercam series, *Sprint*, is a six-degree of freedom tele-operated vehicle capable of autonomous attitude control for remote inspection purposes. It is operated from inside the space shuttle via a UHF radio link and has the ability to autonomously stop at any desired orientation when requested. Astronauts and the ground segment will be sent a stream of video images from a stereo camera pair onboard the *Sprint* platform.

Figure 1.5: The Aercam *Sprint* prototype.

The Aercam *Sprint* has a self-contained propulsion system consisting of twelve compressed gas nitrogen thrusters. The onboard nitrogen tank and lithium batteries can power the free-flyer for up to seven hours, adequate for enduring any current space walk. The compact vehicle also carries an avionics system, two miniature television cameras, a lighting system and a wireless communication module connecting its processor to off-board workstations allowing some high-level control and user interface to be carried out off-board.

The *Sprint* prototype was deployed for a thirty minute test flight in late November 1997 during the STS 87 mission by astronauts Winston Scott, Takao Doi and Steven Lindsay, and the successful flight gave rise to the development of future generations of Aercam at the NASA Johnston Space Centre. These future versions will have an automated navigation and control system including collision avoidance strategies and station-keeping, and will be capable of repair work in addition to inspection applications. Tasks will be delivered using a choice of voice-activated commands, via a hand controller or using a Graphical User Interface (GUI) (*Aercam Objectives*, 2006).

Aercam's control algorithms are based on the *Generalised Voronoi Graph* (GVG) to first find a path through an obstructed space, and then to optimise fuel usage. The algorithm makes use of a geometric *roadmap* which works in a similar manner to transport roadmaps, such that it provides a route to a goal position. This route is formed by connecting the series of locus points of equal distance to two or more convex obstructions, pathing a collision free path. It has three properties: *accessibility*, where a path is found from Aercam's start position onto the roadmap, *connectivity* – travel along the skeletal roadmap, then finally *departability*, the divergence off the roadmap to the goal. The path, once determined, is then optimised, in order to minimise fuel consumption (Choset & Kortenkamp, 1999). **Figures 1.6(a) & (b)** show an artist's impression of an operational Aercam.

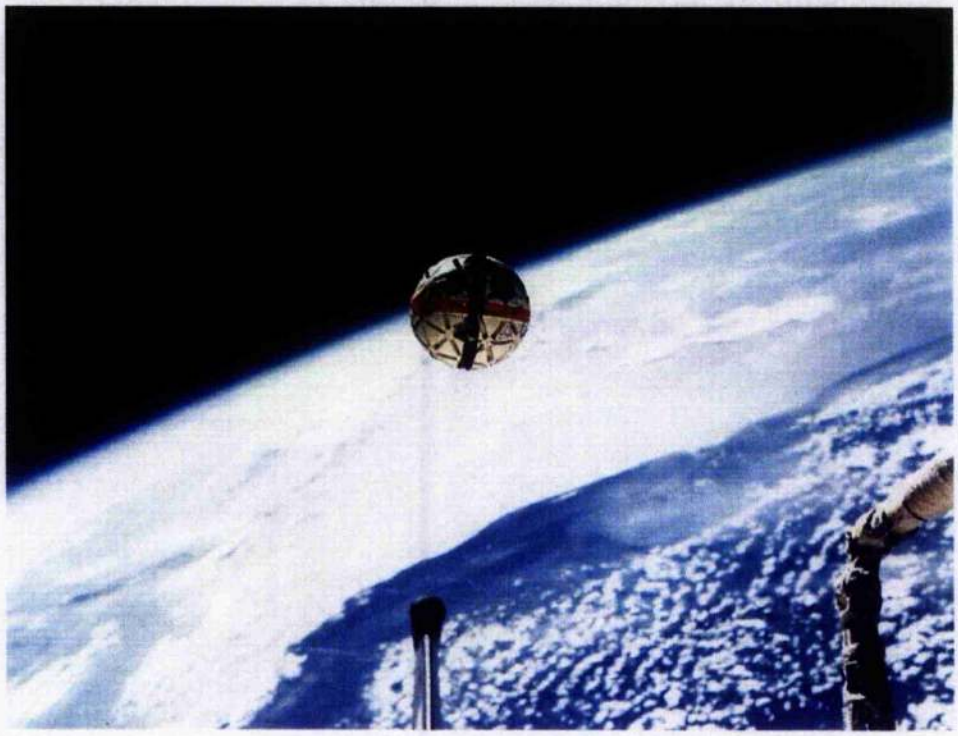


Figure 1.6(a): Aercam in action.

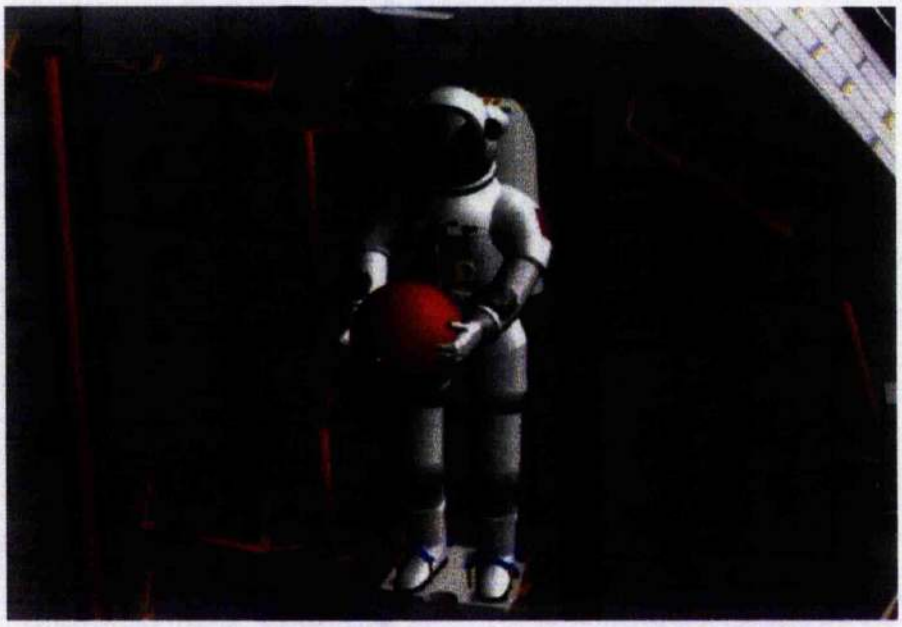


Figure 1.6(b): Artist impression of Aercam.

1.3.4 *Inspector*

Developed by Daimler-Benz Aerospace (now Astrium GmbH), the Inspector observation tool (**figures 1.7 & 1.8**) has been designed conceptually identical to *Sprint* for remote surface observation of the ISS, routine inspection tasks, and EVA support, using a high-resolution camera with autonomous control capabilities (*Inspector*, 2002). The Inspector System is controlled remotely from inside the station and consists of the Inspector Vehicle, the Monitoring and Control Station, and the Transport and Launch Container. The Monitoring and Control Station (MCS) contains a laptop, video display and radio communications, and a navigation system, providing the ground segment with camera orientation and focus. Further control is available from ground-station support through voice, data and video links. Future versions of the Inspector vehicle will be fully autonomous, and will move between target reference points minimising the requirement for manual control.

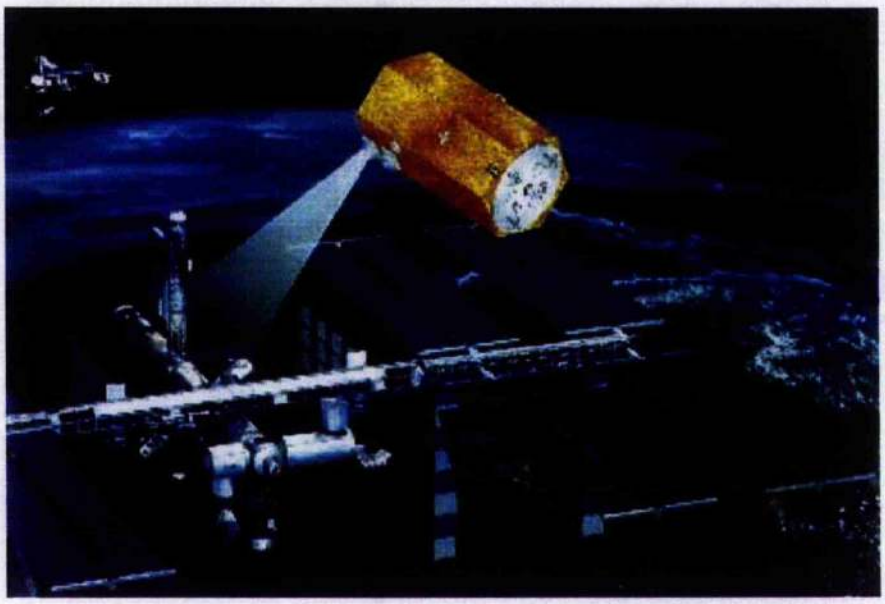


Figure 1.7: The *Inspector.*

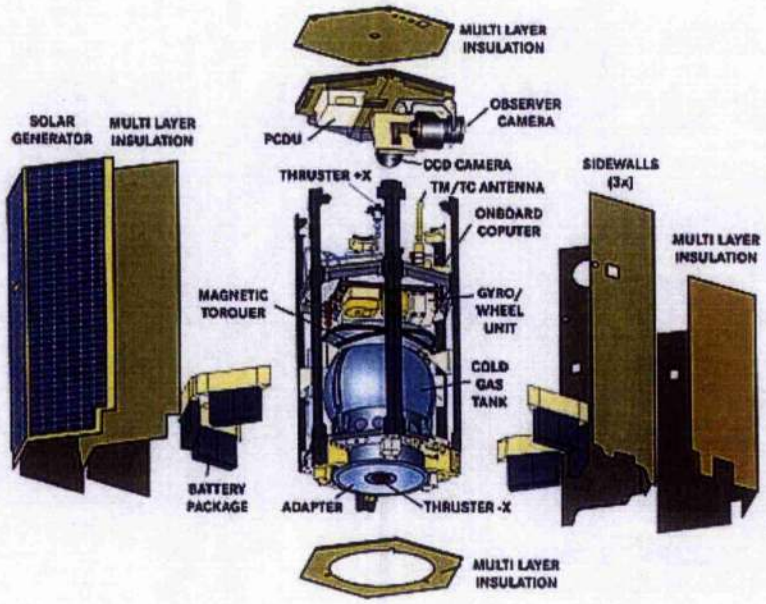


Figure 1.8: The *Inspector* components.

During a mission test in November 1997, the Inspector vehicle was linked to a Progress cargo ship, to be deployed from Mir and injected into orbit. On the first day, the Inspector was to be ejected from the Transport and Launch Container (TLC) fixed inside the Progress docking adapter, and circle once to check functional status. On the second day it was expected to approach and circle the Russian Space Station, as shown in **figure 1.9**, and attempt to transmit detailed pictures of the Space Station exterior. It was hoped to locate damage caused earlier that year when an unmanned cargo ship impacted the Mir Space Station during a practice manual docking. However, shortly after release from Progress, the Inspector malfunctioned and the mission was abandoned. A new Inspector vehicle is planned for use as a robot maintenance tool around the International Space Station for inspection and repair operations.

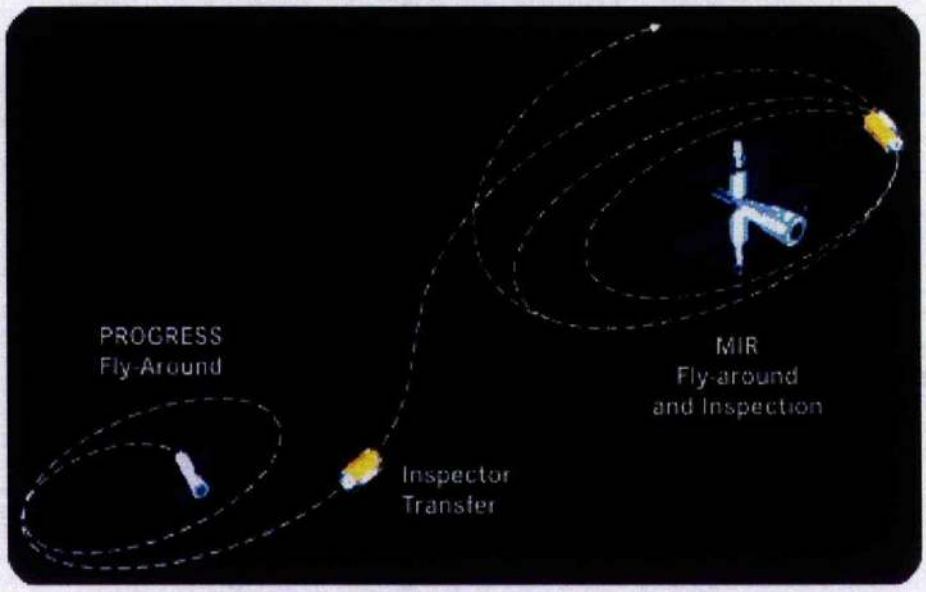


Figure 1.9: The *Inspector* Mission.

1.3.5 *NASA's Personal Satellite Assistant*

For astronaut assistance onboard the International Space Station, NASA is currently developing a free-flying space robot, capable of functioning in micro-gravity conditions within the confines of the ISS. The robot will become integrated as part of the crew and will work alongside astronauts and other autonomous systems providing extra support (*Personal Satellite Assistant*, 2006).

The PSA was evolved from the wireless, data-handling assistant, known as the Personal Digital Assistant (PDA). PDAs are hand-held computer devices that were principally used to enter, store and display data, and were developed at NASA Ames Research Centre by a project team headed by Yuri Gawdiak.

The PDA was initially tested during the Wireless Network Experiment aboard the Space Shuttle Atlantis and the Mir Space Station on March 27, 1996 during the STS-76 mission, and became the first wireless client-server network in space. The wireless networks were discovered to work successfully in space environments, with

no interference to existing avionics, flight computers and communications equipment deployed onboard the spacecraft (*NASA News*, 2002). With a successful outcome, the PDA was then evolved into an intelligent, self-navigating robot, the PSA, illustrated in **figure 1.10**.

This evolved free-flyer will function as a monitoring device for the station, crew and payload by incorporating a number of environmental sensors to measure air pressure, temperature, gas levels and fire detection. It will also be integrated with video and audio interfaces, communication links and electronic support devices to enable remote video conferencing, remote operations support and crew worksite support - performing a vital role as an integrated part of the onboard operations crew, as in the artist's impression in **figure 1.11**.

The free-flyer will be able to collaborate with the station crew by providing an additional set of eyes, ears or information, or perform many of the mundane, repetitive tasks which would be otherwise time consuming to the astronauts, or in potentially hostile or dangerous environments (Dorias, G., & Nicewarner, K., 2003).

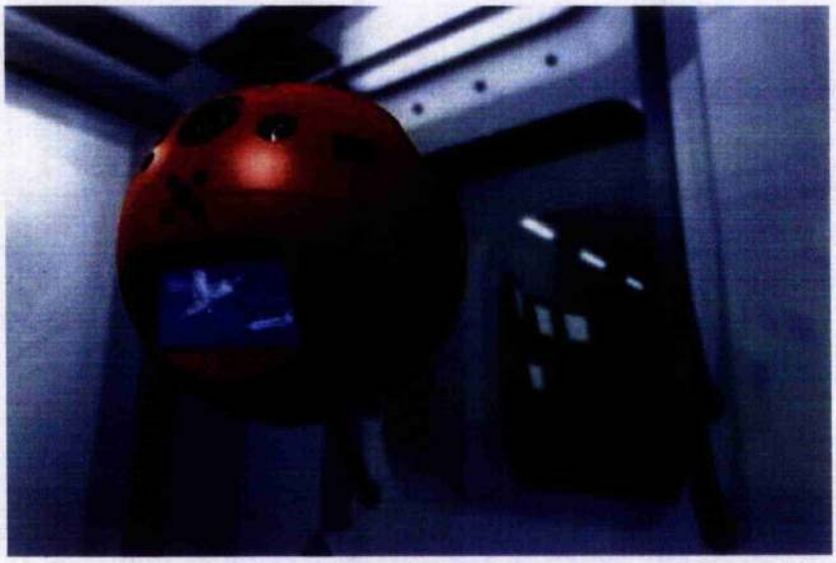


Figure 1.10: NASA's Personal Satellite Assistant

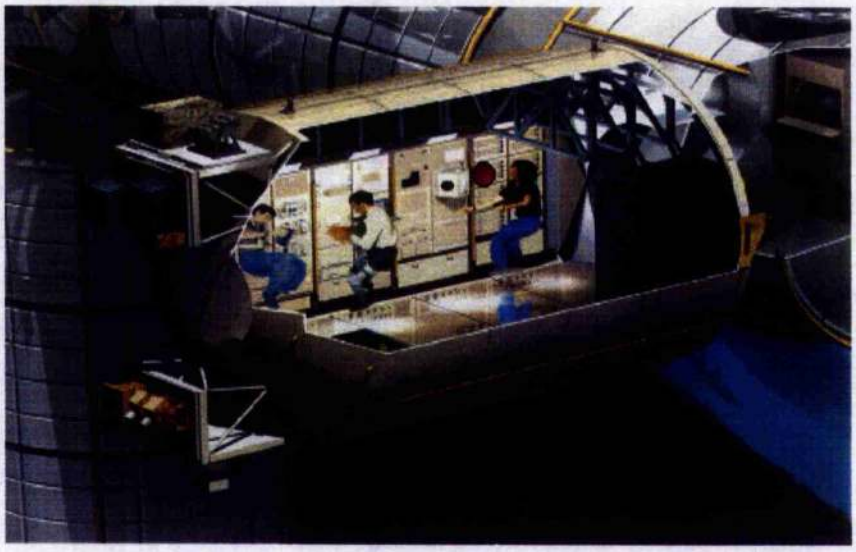


Figure 1.11: Artist's impression of the Columbus Laboratory, with the PSA shown as an integrated part of the onboard crew.

In addition to human-robot collaborations, NASA are also developing co-operations between robots, ensuring autonomous systems of varying capabilities are able to work together to conduct collaborative environmental trouble-shooting. A versatile navigation, control and propulsion system will enable these robotic vehicles to operate autonomously within the spacecraft. Yuri Gawdiak creates the following scenario to emphasize the functions and interactions intended of the free-flyer:

"A crewmember is awoken by a PSA at the requested time. The astronaut asks for a video briefing on the latest events, schedule changes, and priorities while she washes, and eats breakfast. The PSA follows the crewmember through her routine while giving the updates and then checks the inventory database to ensure that the necessary resources are available for the astronaut's first scheduled task. The crewmember logs into her homepage and sets several notifications to be programmed into the PSA to remind her of important activities and times for today's tasks.

As the crewmember works at a payload rack the PSA tracks her movements and provides a remote data terminal capability to allow her to check on procedures

and training instructions, and to support remote video-conferencing and email exchanges with remote colleagues.

Later the crewmember conducts a delicate investigation in the glove box. She requests support from the Principal Investigator (PI) on earth to help her walk through the procedure. The PI calls up a second PSA and manoeuvres about the astronaut and glove box to have an optimum view of the operation and provides real-time feedback to the crewmember. Since the crewmember and the remote PI are absorbed in performing their tasks, the PSA's themselves coordinate the details of their flight and their participation in joint and individual activities without requiring constant attention from their human partners. Moreover, the PSA's are not just passively waiting to be told what to do. They are actively looking for ways to be helpful to the humans in their current task as well as in ongoing responsibilities that have been delegated. For example, as the crewmember uses up supplies the PSA tracks the inventory tags and updates the inventory database.

During a video inspection, a PSA notices that specimens in habitat holding units need food. That evening a pair of PSAs use special integrated payload interfaces and cargo packages to inject supplies such as food into experimental units. One PSA injects the supplies and another collaborating PSA acts as a supply cargo carrier." (Bradshaw, M., Sierhuis, M., Gawdiak Y., 2001)

Awareness of the requirement for these external free-flying observation vehicles has instigated research projects within the University of Glasgow's Aerospace Department. A path-planning tool was developed by Alexander Roger (Roger and McInnes, 1999) to generate safe trajectories for a small, free-flying space robot between an initial docking port and a specified observation point within the vicinity of the International Space Station. In order to be used beneficially by mission planners, long and short-term passive safety checks were maintained during fly around, station-keeping and approach manoeuvres.

Roger's algorithm used Laplace potential functions to enable real-time manoeuvring close to the observation point, where GPS navigation is not plausible due to interference from the ISS structure. This method generates a Laplace potential

field between the boundary volume walls (the volume holding the ISS and its proximity) and obstacle walls (the ISS structure) with a maximum potential field value of 1, whilst the goal point is represented with a potential of zero.

The goal position is found by descending the potential field of the steepest gradient, calculated through a series of iterations (Whyte, 1998). Smooth paths with safe clearance to the goal position are always found using the Laplace potential field function as this harmonic function has no local minima and so will inevitably always reach its goal destination (Liu & Khatib, 2000). However the method requires high computational power, as a result of the large number of iterations required, and any changes to the boundary conditions require a complete recalculation, thus the method is not particularly suitable for changing environments. The algorithm successfully enables safe trajectories to most observation points around the ISS and safe return to the docking port, highlighting the real possibility of having free-flyers operating in close proximity to a crewed space station.

Enhanced mission capabilities allowing object inspection from different view points and positions are made possible by using the Laplace guidance method, to enable observation from points normally difficult to access.

A novel approach to free-flying proximity operations, which would complement the Laplace method, is explored in this thesis. Application of an artificial life (AL) approach, new to space operations, provides the basis for this thesis and allows collision free control of the vehicle in an unknown, changing environment, such as that found within the international space station. An investigation follows into the history of robotic intelligence methods and the benefits of using the A.L. approach. (Khatib, Brock, Chang, Ruspini, Sentis, Viji, 2003)

1.4 A.I. versus A.L.

Throughout the 1960's, the reputed MIT Media Laboratory established the digital foundation for classical A.I. providing a level of machine logic capable of solving a multitude of isolated, complex mathematical problems and complex path planning for robotic systems.

However, there were obvious holes in classical machine intelligence, (with their demand for near human grasps of logic) such as a need for huge computational power to hold complex world models. This discontent for classical A.I. was further reinforced by Daniel Dennett, a cognitive scientist, who, in his paper entitled "Why not the whole Iguana?" (Dennett, 1978), (Girard, Filliat, Meyer, & Guillot, 2005) argued that perhaps the search to find true machine intelligence should be approached from the bottom up. This implies a retreat from a desire for "near-human micro-competences" such as medical diagnosis or chess playing, and instead looks to mimic initially much simpler insect-level animals which exploit the physics of the environment to directly influence their actions (Doty & Bou-Ghannam, 1994). This was a view shared by Hans Moravec in his paper "*Locomotion, Vision, and Intelligence, in Robotics Research 1*" (Moravec, 1984) where he draws to attention the failings of the computer programs devised in the 1960's which provided solutions to mathematical problems in logic, algebra and geometry, could master intellectual games, and "functioned near the epitome of human thought". For whilst there was an initial optimism borne from the success of these programs, future attempts to integrate them into complete robot systems failed miserably, with the subsequent withdrawal of substantial amounts of funding by both the U.K. and USA.

Moravec's belief was that the design of intelligent systems should take lessons from evolution, where our low-level, sensory, locomotive and instinctive control systems are much better developed through longer evolutionary periods, than high level human thought, and as a consequence, is much harder to emulate: "*high level, deep thinking is little more than a parlour trick, culturally developed over a few thousand years, which a few humans, operating largely against their natures, can learn...I argue that the most fruitful direction for this track [the development of artificial life] is along the oxtrail forged by natural evolution.*". (Moravec, 1984)

It was in the early 1990's where a team, lead by Rodney Brooks at the MIT Laboratory used this concept to build 'Genghis' aided by Colin Angle, to challenge traditional robotics. Brooks designed the software, with Angle developing the hardware of this creature.

Genghis was a six-legged robot capable of exploring terrain without the benefit of sight, a complex world model or a central brain. Instead of being trapped in the traditional cognitive bottleneck of perception - building a model of the perceived world - then path planning; the robot's behaviours were based on simple emergent rules with an opportunistic view of the potential interaction between the robot and its environment, predominant in the natural world.

This novel robot was capable of exhibiting deliberate, confident motions including STAND UP, WALK, FORCE BALANCING, LEG LIFTING, PITCH STABILIZATION, PROWLING & STEERED PROWLING providing a rich, exploratory, almost life-like, behaviour. Although Genghis could not beat chess masters at their game nor had any grasp of complex logic or concepts, it could scuttle at speed across rough terrain – an obvious benefit, Brooks remarked, for space exploration.

The intrinsically beautiful aspect of this odd creature, however, was its in-built simplicity. There was no need for high levels of computational power or a vast central processing unit. Instead it was built on a subsumption architecture governed by simple action-selection rules, equipping the robot with the capabilities to successfully manage in the uncertain, ever-changing environment of real life. Brooks argued that this was the only way of allowing silicon intelligence to emerge and evolve, and the only way of allowing these robots to function successfully outside computer simulations or especially designed minimalistic worlds. So was borne a mascot for real Artificial Intelligence, or Artificial Life (Birk, 1998).

Brooks was first inspired in the 1970's to pursue the radical notion of A.L. from a dissatisfaction with A.I. Sharing an undergraduate office at Stanford University with Hans Moravec, who developed one the world's finest A.I. mobile robots to traverse a cluttered room for that time. This robot was equipped with a

cognitive brain and would retain a memory image of the room, recognising specific objects and goals. However, Brooks noted that this robot was terribly slow, computing for fifteen minutes or so before any action was taken. The fundamental A.I. composition of this robot was not efficient, in Brooks' opinion. When working a number of years later at the MIT Media Lab, he was encouraged to set up a mobile robot group with co-workers Anita Flynn and Jonathan Connell to pursue his dream of methodical Artificial Life machines.

Their first creation, a predecessor to Genghis, was named Allen. With Allen, Brooks decided that the current A.I. state of perception- cognition- action should be reduced to only two steps: - perception - action with simultaneous real-time processing of the environment and its state. This was made possible by incorporating an action selection methodology to select the most appropriate behaviour for the robot based on its current state, and as such discards the need for complex A.I. path planning. With this architecture, the behaviour modules were layered, such that lower level behaviours dealt with immediate problems, such as providing collision avoidance strategies, and enabling the robot to function in real-time. Further up, less immediate actions can be taken such as object analysis, terrain exploration or execution of the robot's ultimate goal, for example.

With this layered structure, the robot is designed to work from the bottom up, passing to higher-level behaviours when there are no immediate threats suppressing them. Brook's was confident this methodology would be successful, since many systems in the natural world hold similarities, implementing subsumption, action-selection architectures for their control, He reflects on insects:

"*Insects are not thought of as intelligent. However, they...operate in a dynamic world, carrying out a number of tasks...there may be rain, strong winds, predators and variable food supplies all of which impair the insects' abilities to achieve its goals. Statistically, however, insects succeed. No human-built systems are remotely as reliable.* " (Brooks, 1986)

Brooks was strongly influenced to pursue biological imitations by W. Grey Walter, a neuro-anatomist, and author in the early 1950's, of the intriguing book "*The*

Living Brain" (Walter, 1953). Walter constructed two robots, Elmer and Elsie, resembling turtles, from a series of batteries, tubes and motors concealed underneath a domed shell - a species he named *Machina Specularix* (Reynolds, 2000). The robots embodied several steering behaviours and were designed to seek light sources (illuminating battery chargers) using a random search technique, and were the first machines to exhibit life-like behaviour. The striking manner in which these robots scoured around darkened corners in search of light fascinated Walter to write "*the strange richness provided by this particular sort of permutation introduces right away one of the particular aspects of animal behaviour- and human psychology- that M. Specularix is designed to illustrate: the randomness, freewill or independence so strikingly absent in most well designed machines.*" (Walter, 1953)

Another observation from these robot's emerging behaviour was evident when both robots were seeking to charge their batteries. The strongest robot would succeed, leaving the weaker robot to 'die' as its power became depleted, displaying traits resembling those from Darwin's theory of natural selection (Darwin, 1859).

Walter predicted that in future, these robots would be capable of repair and reproduction, a similar view held by Valentino Braitenberg, a neuro-anatomist working at the Max Plank Institute for Biological Cybernetics, Tuebingen, Germany, some thirty years later. Valentino Braitenberg proposed a series of thought experiments, using hypothetical vehicles functioning in a toy world, constructed from simple sensors and motors, to demonstrate the manner in which intelligence may have evolved through a complex interaction with its environment. In his experiments, what appears to be increasingly complex behaviour emerges from the combinations of very simple algorithms in a multitude of situations, with which he draws parallel to the synaptic activities of the cerebral cortex through their simplicity and regularity.

He describes these artificial machines as though animals in their natural environments, and draws comparisons to bring to attention the uncanny likenesses that emerge, even though, he points out "*There is nothing in these vehicles that we have not put there ourselves*" (Braitenberg, 1984).

His simplest vehicle 'V1' has one sensor and one motor connected such that the speed of the motor is proportional to the signal level detected by the sensor. If exactly proportional to absolute temperature, the vehicle will slow down in cool regions and speed up in hot regions. By adding the effects of environmental friction, he observes the vehicles motion becomes erratic (but with direction), due to the microscopic, unsymmetrical frictional forces acting on the motor. "The environment," he explains, "can have a powerful effect on how the vehicle is perceived. The vehicle is perceived as being restless and disliking warm water. In any case, it is said to be ALIVE."

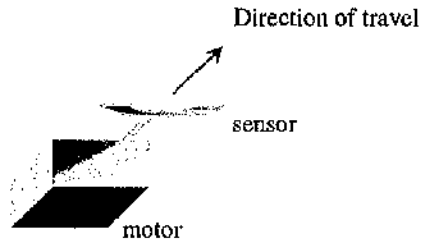


Figure 1.12: Vehicle 1: Levels detected by sensor determines speed of motor.

Progressing from the first vehicle, the second vehicle contains two sensors and two motors. Again, in this experiment, the speed of the motors is proportional to the signal level detected by the sensor. There are two interesting variations of this vehicle depending on the sensor-motor connections:

(a) Each sensor is connected to the motor on the same side.

(b) Each sensor is connected to the motor on the opposite side.

(a) (b)

Figure 1.13: The two variations of Vehicle 2

Vehicle 2(a) displays the behaviour of **COWARD**, as it dislikes its source, and tends to turn away from it. (The wheel closest to a light source shall spin faster than that furthest from the source.)

Vehicle 2(b) may be described as **AGGRESSIVE** as it turns towards its source, hitting it at speed. (The wheel furthest from a light source spins faster than the other.)

Braitenberg then goes on to expand his range of machines, placing them in different environments, and with different sensors and sensor/motor connections, and notes the interesting behaviours that emerge (Braitenberg, 1984).

By varying the sensors and motors and their connections, he discovers an extraordinary range of anthropomorphic behaviours characteristic of cowardice to love. And, following this, Braitenberg suggests that if these artificial animals or 'animats' (Wilson, 1991) (Maes, 1991) behave as though they are alive, and exhibit emergent, unprogrammed behaviour, they may well be considered to be as such. Perhaps there is not a world of difference between the forces that drive them:

"Although there is nothing in these machines we have not put there ourselves, there is nothing in complex cellular automata not determined by the rules, nothing in humans but what is in the embryo, and nothing in the entire spectrum of life not evolved by a single cell...how could we be so sure the behaviour from these wired machines was so qualitatively different from behaviour in the natural world" (Braitenberg, 1984)

Although the hardware and the environment of Braitenberg's experimental vehicles may not be the same, the principle has been adapted in the work of this thesis for the development of a robot controller for application within a space environment. It shall be shown herein that the adaptation of Valentino Braitenberg's principles can form a useful control methodology for a robotic vehicle, providing the basis for the intelligence required for collision avoidance, goal searching and refuelling.

Braitenberg argues that his range of vehicles may be capable of exhibiting conscious decision-making, or the possible implications of freewill, and moreover provides a multitude of examples supporting the law of uphill analysis/downhill invention, a term commonly used to explain the concept of a machine which appears outwardly to have a very complex make-up, but which, in fact, is relatively simple given knowledge of that make-up. It is easy to design a machine capable of complex behaviour seemingly governed by emotion, however it is more tasking to determine the mechanisms that govern these behaviours and how they come to arise. For example, Braitenberg devises a vehicle whose motor speed is dependent on the intensity of stimulation. However, by incorporating sensor efficiencies and thresholds in motor activation, the behaviour patterns of the vehicle are notably different.

"These creatures, the observer would say, ponder over their DECISIONS. When you come close to them with a lure, it takes them some time to get going. Yet once they have decided, they can act quite quickly. They do indeed seem to act in a spontaneous way... you would almost be tempted to say: where decisions are being made, there must be a WILL to make them. Why not? For all we know, this is not the worst criterion for establishing the existence of free will. "

"These creatures, the observer would say, ponder over their DECISIONS. When you come close to them with a lure, it takes them some time to get going. Yet once they have decided, they can act quite quickly. They do indeed seem to act in a spontaneous way... you would almost be tempted to say: where decisions are being made, there must be a WILL to make them. Why not? For all we know, this is not the worst criterion for establishing the existence of free will."

He points out that it is difficult to analyse the internal mechanisms of these vehicles purely from observation of their behaviour, since, many mechanisms have identical behaviour, and we have a tendency to overestimate complexity.

Braitenberg's concept was the inspiration for the development by the MIT-Media Laboratory of a series of autonomous vehicles implemented in LEGO-LOGO. These were programmable, self-contained bricks capable of imitating animal behaviours using a perception/action concept, and eliminating the need to download from an external computer. MIT-Media laboratory's Mitchel Resnick (Resnick, 1989) presented an experiment at an artificial life workshop in the late 1980's, however, it became apparent that aside from carrying out the task expected, a degree of unprogrammed, emergent behaviour was found to arise. The project entailed writing a program that would enable the LEGO brick to follow a line along the floor. Unexpected behaviour emerged when the brick reached the end of the line. No part of the program was designed to deal with this situation, but much to his surprise, the robot turned and followed the line in the opposite direction. This exciting and important concept of unprogrammed, emergent behaviour has pathed the foundations of A.L. in many fields of today's technologies.

Emergent behaviour arising from a response to animat state was found to occur from experimentation by Michael Travers, a Media Laboratory graduate, on artificial ants. The 'ants' would search for a food source, and when found, return to the 'nest'. On return, as a consequence of holding food, the ant would leave a pheromone trail. Emergent behaviour arises since the pheromone trail becomes a track to other ants to the source of the foodstuff (Travers, 1988). Randall Beer, a researcher at Case Western Reserve University, also adopted an interest in emergent behaviour, working on the neural structures of insects. He constructed a computer generated artificial

insect whose locomotion was based on the American Cockroach. Its six legs moved in such motion as to maintain a stable gait by controlling its centre of gravity. He decided to investigate the repercussions of severing the motor neural connections to one of the cockroach's six legs. Much to his surprise, he found the digital insect could maintain its stability by converting to an alternative gait. By altering the neural activation levels and firing speeds, five different, unprogrammed gaits emerged, all of which have been recognised in real insects (Beer, 1990).

Brooks considered the possibility of using these insect-like robots for space exploration, but faced the problem of time delay. Relaying information from Earth to space to control the robot-insect's actions would be impractical, due to the lengthy time for data transfer. It would be necessary, for adequate response times, for the robot to be independently operational and not have to rely on signal responses from Earth to other celestial bodies. Brooks, when commissioned by NASA to develop a potential Mars explorer, to preceed a possible manned mission, looked to develop and adapt his Genghis. He argued however, that a robotic mission should look to release hundreds of miniature, six-legged, robots on the Mars surface, to increase survival chances, rather than one 'dinosaur' which would be doomed to failure if a fault occurred. It would be of minimal loss to loose part of the swarm of insects built at, comparatively, a fraction of the cost of a single large rover. These small robots would be capable of acting independent of external control, and would communicate within their group, to facilitate the shared goal. It would be of no great cost to loose part of the colony through faults or defects, and by implementing a bottom-up, behavioural approach they would benefit from unexpected environmental occurrences. Brooks developed, with his colleague Anita Flynn, *Squirt*, the first prototype gnat robot, a robot postulated to be miniaturised to barely the size of a ten pence piece by incorporating silicon micro-motors and microscopic circuit boards (Brooks, 1990). Flynn envisioned swarms of these tiny robots released from a central orbiter, to explore the surface of a planet and transmit back sensory information (Flynn, 1987). This was a vision also anticipated by Pattie Maes.

Pattie Maes was an assistant professor of computer science at MIT's Media Lab, who designed Genghis' walking algorithm and was interested in the potential benefits of using emergence borne from swarms of artificial life machines. She was

frustrated at the lack of funding from agencies holding the view that animals were sub-optimal. The agencies viewed animals as displaying inefficient robotic behaviour, and so tended to support more traditional methods of AI rather than those which following natural methodologies. She welcomed ethology, however, disillusioned with robots that were computationally restricted by a need to hold an entire *world* in their heads. And so, when on sabatical to the University of Brussels, she conducted experiments with Luc Steels, a professor at the artificial life laboratory, on the potential power of emergence of *adaptive autonomous agents* (Steels, 1994(a)) - systems that try to complete a set of goals or motivations using its own resources to operate independently in an unpredictable environment (Maes, 1991), (Steels, 1994(b)). They explore the use of autonomous agents strongly inspired by animal behaviour, resulting from the interaction between the agent and its environment. By developing prototype LegoTechnics robots equipped with sensors and motors, they conducted a number of experiments to highlight the potential benefits in the new field of behaviour-based AI. And further, to highlight emergent behaviour arising from multi-agent systems (an ecosystem containing two or more autonomous agents) when no explicit co-operation was programmed. This led to robust working systems capable of survival in unpredictable environments, an idea shared by Keith Doty working at the Machine Intelligence Laboratory at the University of Florida, and Akram Bou-Ghannam, employed by IBM. Doty and Bou-Ghannam insisted that a complete control architecture would employ behavioural, cognitive and perceptual components, running in parallel to provide instinctive and knowledge-based intelligence. They point out that animals lower in the hierarchy rely mostly on reactive, instinctive behaviour, whilst on climbing the hierarchy there is a greater reliance on learning, reasoning and planning. They argued that the most resilient intelligent robotic system would be one employing both cognitive and reactive modules (Doty & Bou-Ghannam, 94).

Pattie Maes pointed out a number of distinctions between classical and behaviour based AL methodologies:

- Traditional AI concentrates on developing isolated, specific, complex competencies whereas behavioural AL focuses on multiple, broad ranging, lower level, competencies.

- Traditional AI maintains no direct interaction with the environment, except via a human operator - closed system. An autonomous agent is, however, situated in its environment and is directly influenced by the environment, detected through use of sensors, and responding through actuators- open system.

- Traditional AI deals with only one problem at a time, whereas an autonomous agent works in a dynamic, ever-changing environment, often dealing with multiple conflicting goals within a given time restraint.

- Traditional AI is knowledge-based, and static, in that it is only active when a problem is put to the system. Autonomous agents are, however, behaviour-based and dynamic, always adjusting to their environment.

- Traditional methods do not adapt, such as with component breakdown, in contrast to new methods of AL where adaptation and improvements are fundamental.

This optimism in Artificial Life approaches is shared by Daniel Dennett in his paper "*Intentional systems in Cognitive Ethology*", where he remarks " *I suggested that people in AI could make better progress by switching from the modelling of human micro-competences (playing chess, answering questions about baseball, writing nursery stories, etc.) to the whole competences of much simpler animals. At the time I suggested it might be wise for the people in AI just to invent imaginary simple creatures and just solve the whole mind problem for them. I am now tempted to think that the truth is apt to be both more fruitful, and, surprisingly, more tractable, than fiction. I suspect that if some of the bee and spider people were to join forces with some of the AI people, it would be a mutually enriching partnership.*" (Dennett, 1983).

Chapter Two: Model Definition and Development

'Elephants don't play chess'

Rodney Brooks

2.1 Introduction

In order to test the control methodology constructed within this thesis, a virtual test-bed must be developed, within which all the control algorithms may be tested, and from which performance data may be extracted for analysis. The test-bed was composed of a three-dimensional virtual environment, representing a space station module, complete with potential collision hazards, a refuelling station, and a goal base. The environment was generated using the OpenGl® software interface, (Woo, Neider, Davis & Shreiner, 1999) with the coding for all control algorithms produced using the C programming language (Sexton, 1997).

2.2 Constructing the Environment

OpenGl is a hardware-independent, software interface capable of use on numerous hardware platforms for the production of interactive three-dimensional applications. There are no high-level commands for the construction of complicated models, and as such, the desired model has to be rendered from primitives: polygons, lines, points and bitmaps. There are, however, libraries which can be built upon OpenGl® to allow the rendering of more complicated surfaces, such as the GLUT library (OpenGl Utility Toolkit), which includes routines to create more complicated objects e.g. spheres, cones and cylinders. OpenGl® includes a number of features to enhance the image rendered, such as the inclusion of atmospheric effects (e.g. *fogging*, to give the appearance of depth), lighting, colour, antialiasing (to smooth jagged edges), shadowing and textures. The points, lines and polygons, which define

the image, are constructed from their vertices and are displayed as pixels on the screen.

Figures 2.1 through **2.8** show how the virtual environment was built up to represent two inter-connecting space station modules, and as such, provide a test environment for the controller developed in later chapters. **Figures 2.1** and **2.2** show the initial 3-dimensional wire-mesh skeleton of the space station modules, during its initial stages of construction (**figure 2.2** is a close-up of the lower of the two modules). The modules were rendered by plotting the vertices of the polygons making up each of the faces. The small sphere located in the top right hand corner of **figure 2.2** is the representation of the free-flying robot. In **figure 2.3**, the wire-mesh polygons have been filled, a colour for each polygon, to show a flat-shaded, unlit version of the modules. They appear flat, since only one colour has been used to fill each polygon (each wall). There are no light source effects.

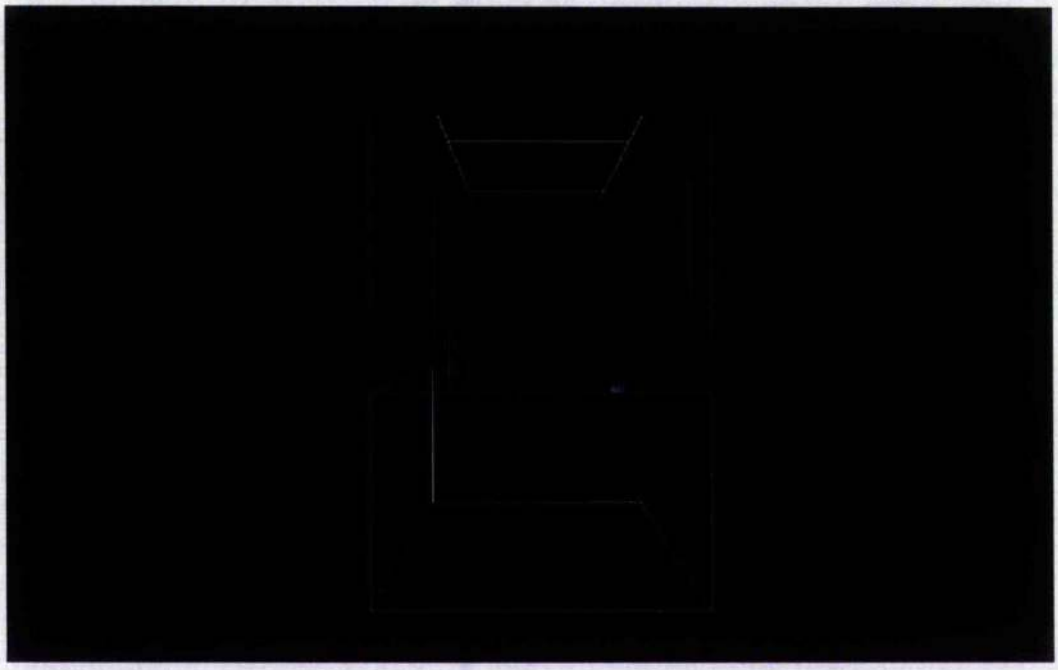


Figure 2.1: View of station modules during their initial construction phase.

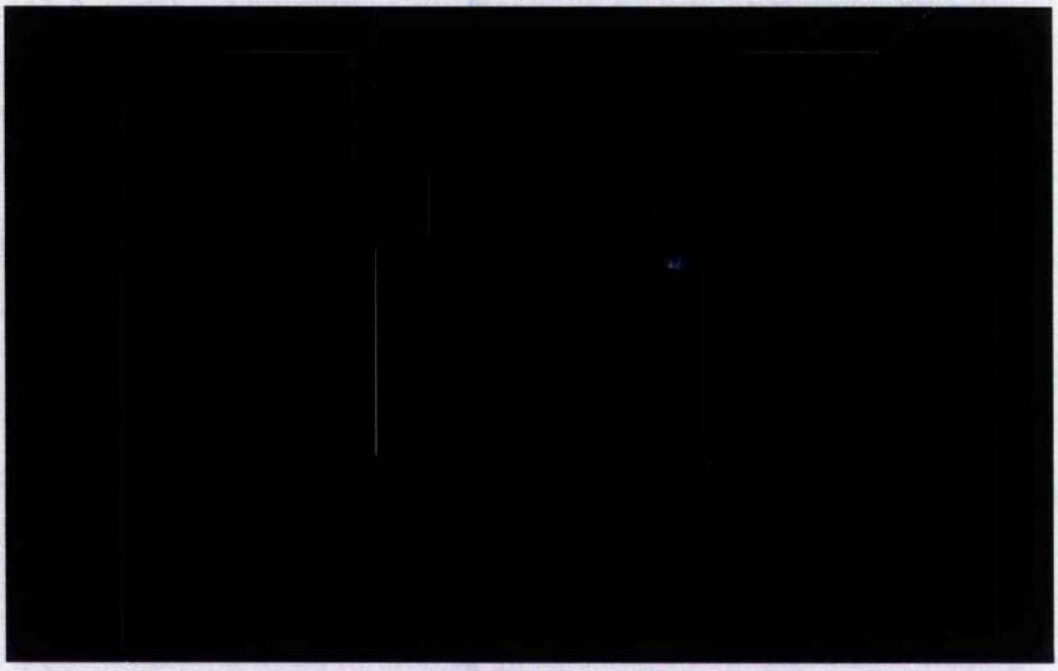


Figure 2.2: View of the lower of the two space station modules.

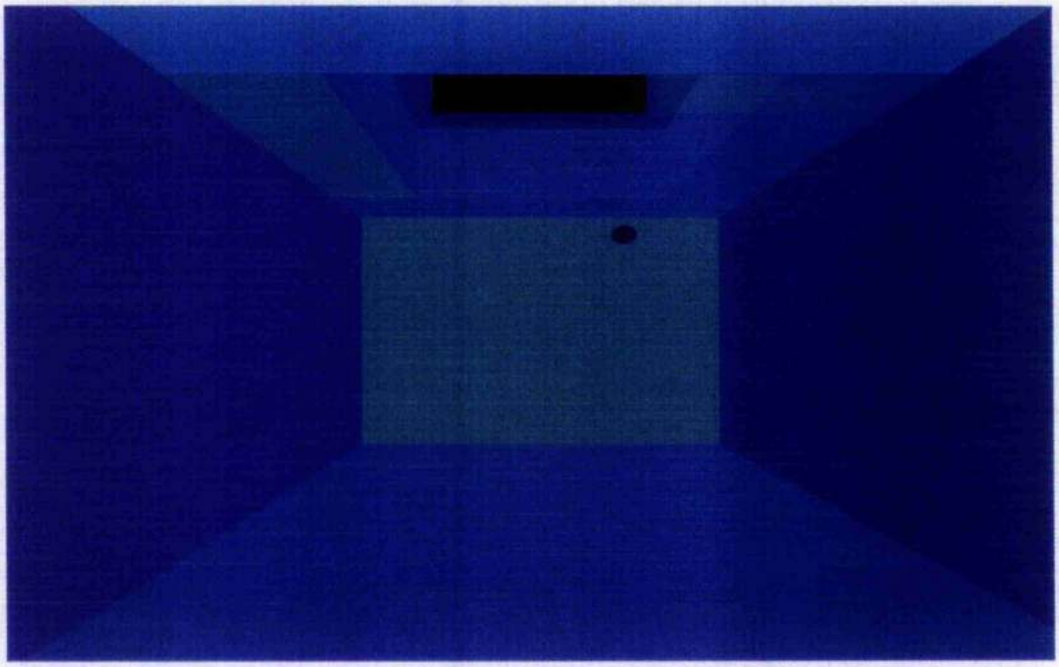


Figure 2.3: Snapshot of the lower module with flat-shaded polygons.

In **figure 2.4**, the effects of lighting have been added (with uniform wall colouring) making the module more realistic and three-dimensional, shaded as though in response to the light source emitting from the far left-hand corner. In addition, two coloured triangles have been included in the environment, these are markers of the positions of the refuel depot, and goal base.

Figure 2.5 sees the incorporation of some collision hazards in the form of a hypothetical circular control station and railing, and in **figure 2.6** and **2.7** windows have been cut into the module, displaying a textured view of space in all directions. This was made by encapsulating the space module within a large outer cube with textured surfaces. The refuel depot and goal markers have been replaced with a stand and platform, and the series of blue spheres represent positions of possible goal points, for future robotic tasks (for example, inspection of possible fluid or gas leakage). **Figures 2.8** and **2.9** display the external view of the space station module, complete with lighting effects and solar panels, whilst **figure 2.10** displays the interior in wire mesh mode to show how the module has been built up.

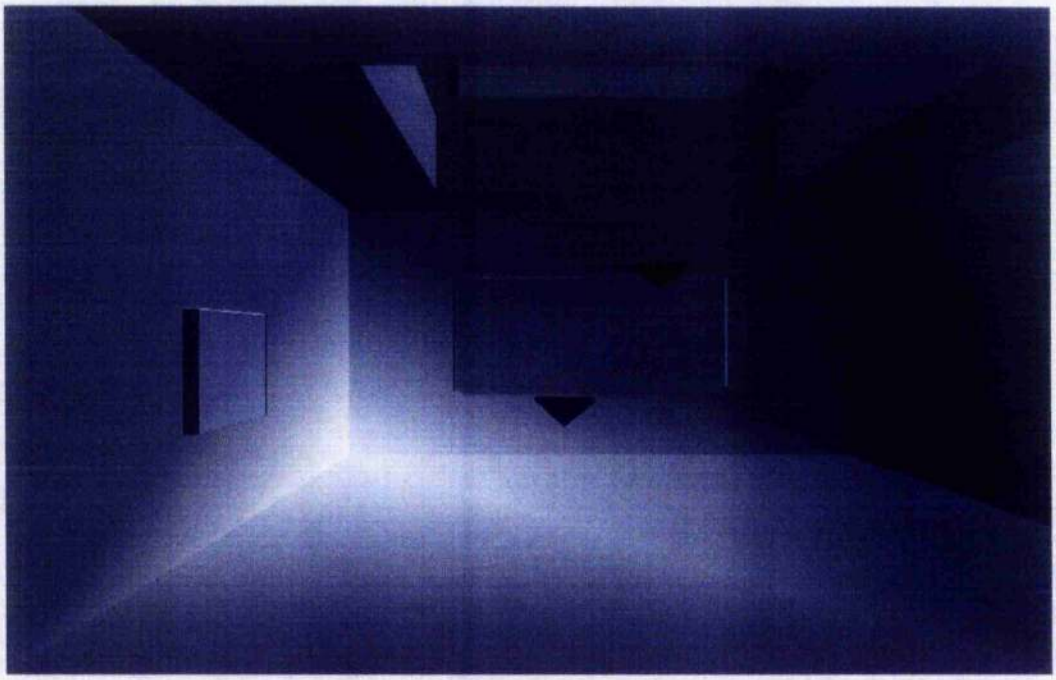


Figure 2.4: Lighting adds depth to the environment.

Figure 2.5: Potential collision hazards included within space modules.

Figure 2.6: Environment now displays windows with textured space view.

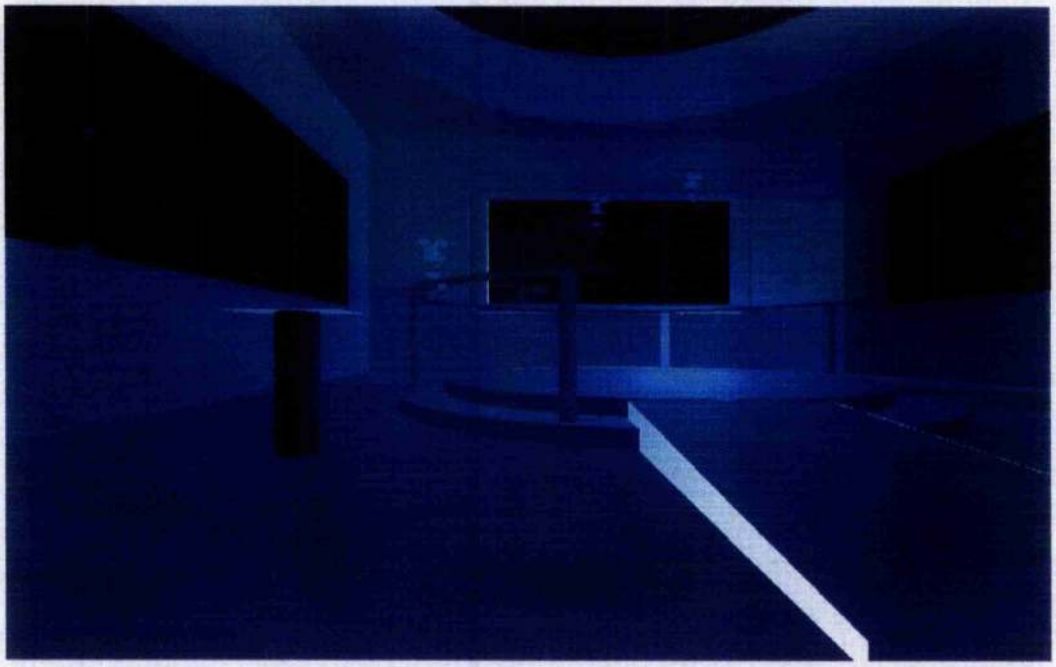


Figure 2.7: Completed environment displaying textured background. Bubbles represent positions of potential gas leaks.

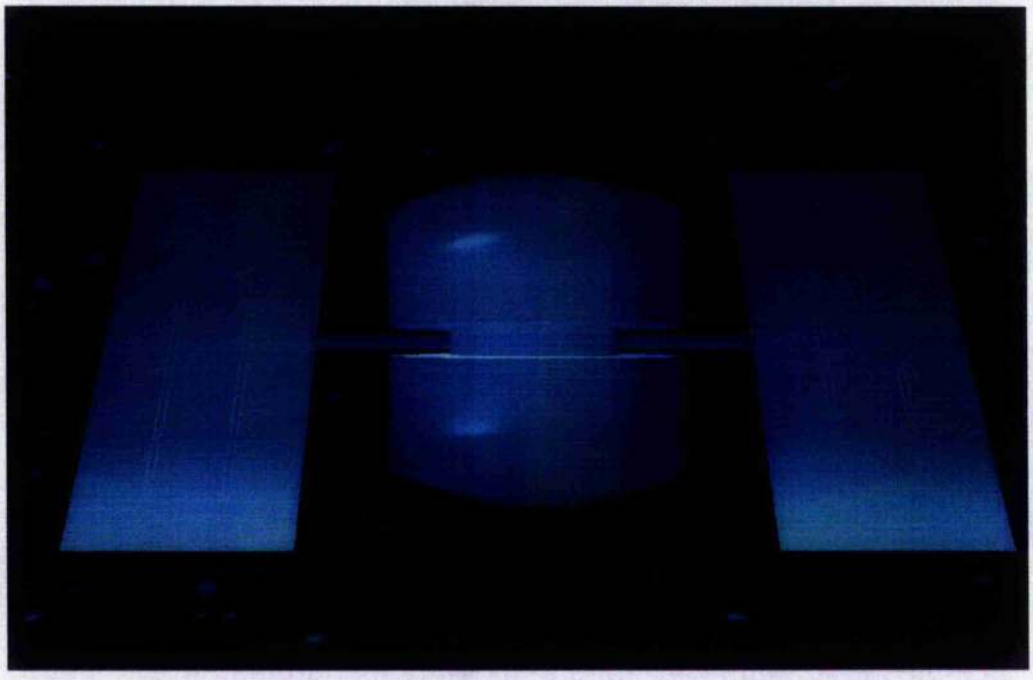


Figure 2.8: Exterior view of the space module, complete with solar panels.

Figure 2.9: View of space module from a distance.

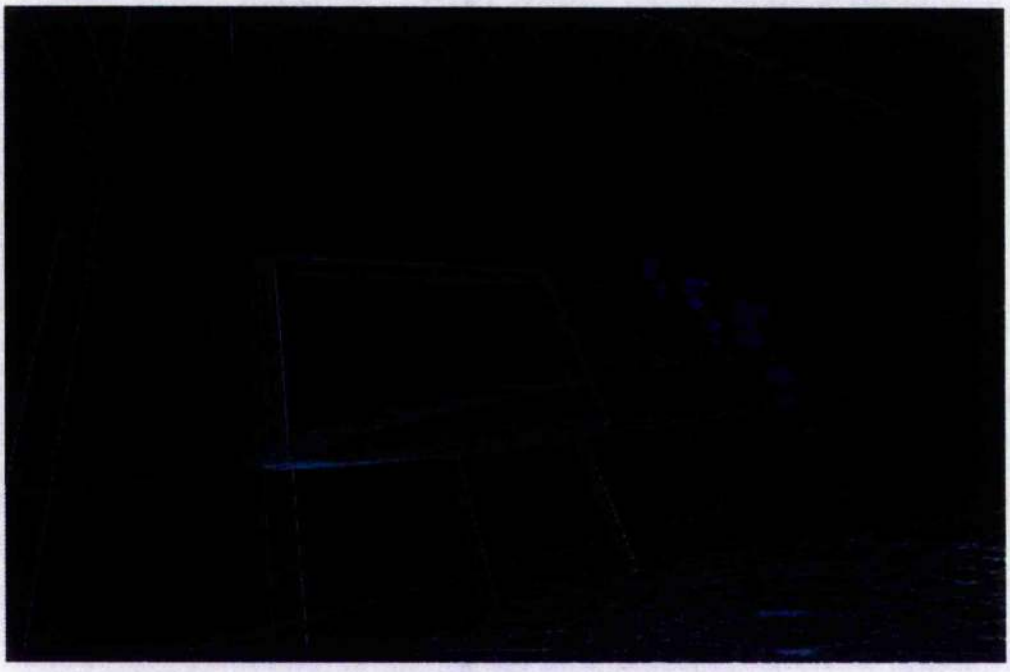


Figure 2.10: Interior of space module in wire-mesh mode.

2.3 Free-flying robot: design specification

The robot, as simulated in **figure 2.11**, is defined as a small, 2.5 kg, spherical unit with cold gas thrusters for actuation. The maximum velocity is taken to be 0.1 ms^{-1}, a ceiling acceleration of 0.2 ms^{-2}, and a Specific Impulse of 50 s. The following specifications are here listed:

Mass (m_r)	2.5 kg
Radius (r_r)	0.15 m
Moment of Inertia (I) = $2/5 m_r r_r^2$	0.0225 kgm^2
Maximum velocity (V_{max})	0.1 ms^{-1}
Maximum acceleration (a_{max})	0.2 ms^{-2}
Propulsion Specific Impulse (I_{sp})	50 s

The design parameters have been chosen reflecting those of the NASA Aercam (*Aercam*, 2002), or the proposed NASA Personal Satellite Assistants (*Personal Satellite Assistant*, 2002).

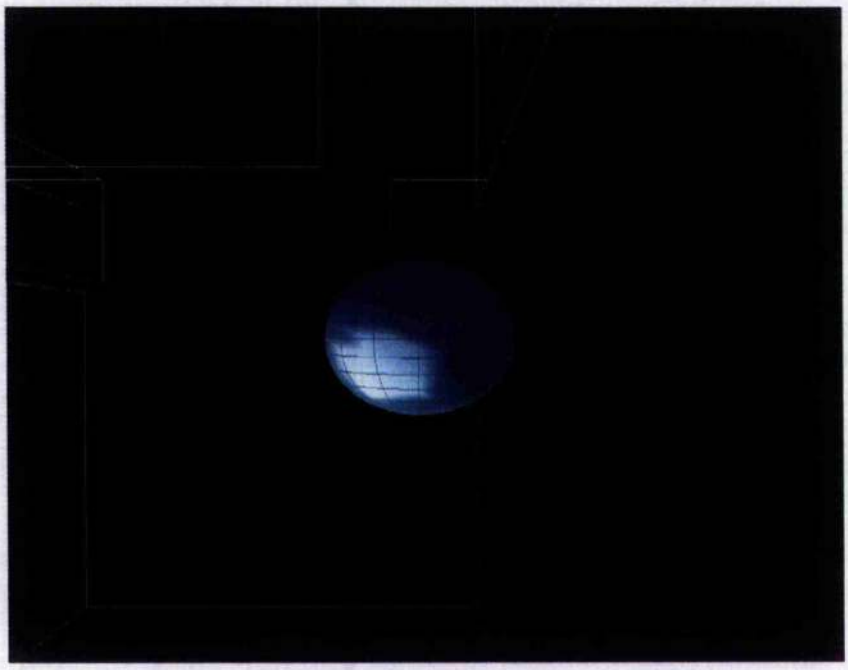


Figure 2.11: Simulated free-flyer using OpenGl®.

2.4 Free-flying robot: design application

To determine the location of the robot in its environment requires either an accurate geometric model of the environment a priori, which requires significant computational power and necessitates that the environment remains unchanging, or alternatively, the use of sensors to measure distances between the robot and its environment directly. This latter approach is taken herein for these reasons, and in order to follow the principles of Braitenberg's behavioural methods. All contact with collision hazards could be eliminated if perfect measurements could be taken from every point on the surface of the robot's structure. However, since this is impossible, no sensor is perfect, and only a finite number of sensor points may be taken, then it demands that the best choice of sensor, governed by data quality, coverage, cost and safety, become the pertinent issue.

The sensor technologies commercially available include amongst others: laser triangulation, ultrasonic, inductive or capacitive. With laser triangulation, a narrow beam of laser light is projected, and the measurement of the location of the reflected light measured at an angle, determines the distance. This type of sensor has a very short range and to increase the range requires increasing the strength of the laser to those which are not eye safe. Since eye safety is an important issue for space robots used on crewed spacecraft, the use of lasers is eliminated. The inductive and capacitive sensors are mainly designed for very short-range assembly line detection and are not appropriate for a free-flyer, requiring a range of more than a few centimetres due to potential collisions at speed. Ultrasonic sensors are lightweight, cheap and harmless, and have reasonable range, and whilst useless out with the space station (due to the presence of a vacuum) are adequate for navigation within the station itself. These position sensors detect the presence of objects within the supervised range by periodically sending out a short, intensive sonic impulse (Volpe & Ivlev, 1994). As shown in **figure 2.12**, this impulse is partially reflected on meeting an object in its path and the echo returns to the sensor, which then computes the distance from the time interval between sending receiving the impulse (Welotec, 2003).

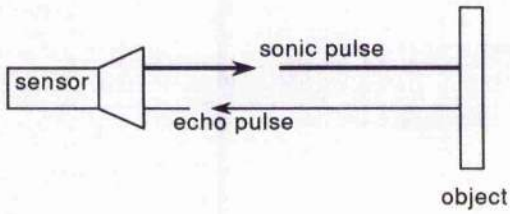


Figure 2.12: Ultrasonic sensor function.

Navigation information is therefore drawn from 6 ultrasonic, proximity sensors placed around the free-flyer's body, two of which lie along each of the free-flyer's 3 orthogonal axes. The information from these sensors motivates behaviours to attain set objectives such as, in the first instance, obtaining collision-free motion perhaps whilst also performing a useful function.

In order to simulate the information obtained from the proximity sensors, the environment is mapped onto a three-dimensional mesh array (see **figure 2.13**). Within this array, any coordinate point lying in free space is given a value of zero, and obstacles and walls are given a boundary value of 1.

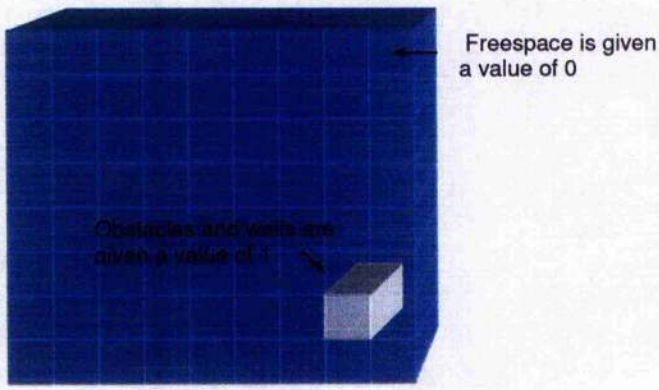


Figure 2.13: Mesh grid onto which 3-D environment is then mapped.

Figure 2.14 displays a portion of pseudo-code, to show how the lower module of the space station (dimensions 16x16x35) is initially coded to give free-space a value of 0, and **figure 2.15** the pseudo-code giving walls (which lie along boundary x = y = z = 0, x = y = 15, z = 35) a value of 1:

```
#define length-of module-in-x-direction-from-zero 15
#define length-of module-in-y-direction-from-zero 15
#define length-of module-in-z-direction-from-zero 34
int boundary[16][16][35];    ///this defines the mesh grid dimensions

void set_boundary()
{
int i=0;
int j=0;
int k=0;

for(i=0; i<=xrange; i++)
{
        for(j=0; j<=yrange; j++)
        {
                for(k=0; k<=zrange; k++)
                {
        boundary[i][j][k]=0;    /// this sets every point within
                                the mesh grid initially to zero
                }
        }
}
```

Figure 2.14: Pseudo-code sets all coordinates within mesh grid to zero (free-space).

```
for(i=0; i<=xrange; i++)
{
        for(k=0; k<=zrange; k++)///this sets the boundary walls
                                    on the y-plane to a value of 1.
        {
boundary[i][0][k]=1;
boundary[i][yrange][k]=1;
        }
}


for(j=0; j<=yrange; j++)
{
        for(k=0; k<=zrange; k++)    ///this sets the boundary walls
                                    on the x-plane to a value of 1.
        {
boundary[0][j][k]=1;
boundary[xrange][j][k]=1;
        }
}

for(i=0; i<=15; i++)
{
        for(j=0; j<=15; j++)
        {                           ///this sets the boundary walls
                                    on the z-plane to a value of 1.
boundary[i][j][0]=1;
boundary[i][j][34]=1;

        }
}
```

Figure 2.14: Pseudo-code sets all coordinates within mesh grid to zero (free-space).

As obstacles are introduced into the environment, they are likewise given boundary values of one. The 6 virtual sensors operate by each counting along its line of sight until such time as the boundary value recorded changes from zero to one (hence having detected an obstacle), *or*, until having reached its sensor horizon, whichever comes first. The sensor horizon is the maximum distance at which a sensor can still determine distance. In **chapter 3**, a range of sensor horizons are examined, typical for ultrasonic sensors, between 0.5 metres and 8 metres showing the influence this horizon has on the robot's trajectory.

The propulsion system, enabling it to operate autonomously throughout the space-station, is composed of a pressurised nitrogen tank supplying 12 compressed nitrogen gas thrusters composed of a simple on/off valve and nozzle, 4 of which lie along each of the free-flyer's 3 orthogonal axes, as shown in **figure 2.16**.

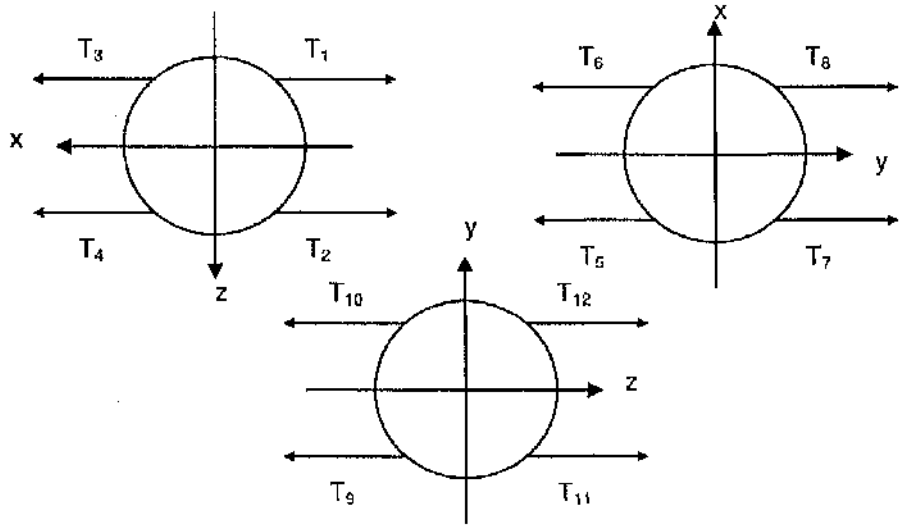


Figure 2.16: Thruster locations on free-flyer.

Thrusters T_1 to T_4 lie in the y=0 plane parallel to the x-axis, and enable either translation parallel to the x-axis (using thrusters T_1 and T_2 together, or T_3 and T_4 together), or, rotation around the y-axis (using thrusters T_1 and T_4 together, or T_2 and T_3 together). Likewise thrusters T_5 to T_8 lie in the y=0 plane parallel to the y-axis, and enable either translation parallel to the y-axis (using thrusters T_5 and T_6 together, or T_7 and T_8 together), or, rotation around the z-axis (using thrusters T_5 and T_8 together, or T_6 and T_7 together). And finally, thrusters T_9 to T_{12} lie in the y=0 plane parallel to the z-axis, and enable either translation parallel to the z-axis (using thrusters T_9 and T_{10} together, or T_{11} and T_{12} together), or, rotation around the x-axis (using thrusters T_9 and T_{12} together, or T_{10} and T_{11} together).

Other possible alternative forms of propulsion system suitable for an onboard free-flyer could include an electrically powered motor/ propeller system. This would have the advantage of being less cumbersome and less complex in terms of refuelling (filling compressed nitrogen tanks versus plugging in to a mains electricity supply), however, with the electrically driven motor alternative, the magnetic current generated around the free-flyer could affect the operation of sensitive onboard equipment. Furthermore, with nitrogen thrusters, the ease of calculation of the propellant consumption used allows for determination of when to refuel, which helps in later behaviour simulations. For these reasons, it was decided to use the nitrogen propulsion system.

As discussed in **Chapter 1**, the free-flyer acts as an astronaut assistant, equipped with a camera for video conferencing or web material, and may be further equipped with a variety of sensors to monitor environmental conditions in a spacecraft such as the amount of carbon dioxide, oxygen and other gases in the atmosphere, the amount of bacterial growth, air temperature and air pressure. Tracking astronaut motion is most likely to be carried out using image processing from a digital camera.

2.5 Acceleration Control and Velocity Control

The control methodologies explored in this thesis in **chapters 3** and **4** fall into two categories. The first, *acceleration control*, ignores the robot's attitude control,

and merely focuses on manoeuvring the robot around obstacles without altering the orientation of its body axes. Its component velocities are altered through integrating incremental changes in acceleration as governed by a function of distance measured (by the virtual sensors) to potential collision hazards, as shown in **figure 2.17**. Here, thrust acts through the robot's centre of mass and, as no moment is thus produced, the trajectory formed is linear. This is demonstrated in **chapter three**.

The second category, the *velocity control* methodology (**figure 2.18**), makes use of the robot's rotational capabilities to steer the robot around obstacles by virtue of manipulation of the robot's velocity vector. By producing a moment around the free-flyer's centre of mass, and hence rotating its body axes, a smoother, more versatile trajectory is obtained, which has clear benefits in terms of camera pointing versatility explored in later chapters. Since the model is built up in increasing degrees of complexity, however, the design starts with the simplest method in **chapter 3**, before advancing, in **chapter 4**, to velocity control methods.

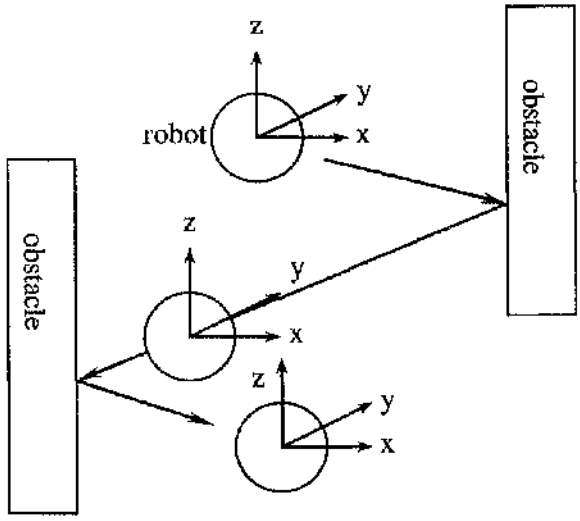


Figure 2.17: Acceleration control system

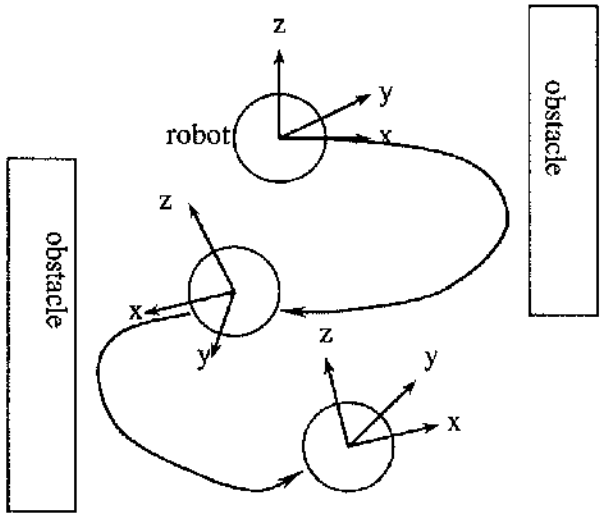


Figure 2.18: Velocity control system

Chapter Three: The Acceleration Control System

'These machines, they're, they're pure evil...
we've gotta stop them somehow, godammit,
before they take over the earth and I think I have a plan.
This may be crazy talk, but it might just work...'

Satan's Satellites

3.1 Introduction

Having established in **chapter 2** the two control systems to be explored, this chapter delivers an in-depth description of the development of the first of these systems, namely, the acceleration control system. The second system is discussed in **chapter 4**. Ignoring attitude control for the present, the construction of this simpler, inertial system provides an initial basis for analysis, and systematic prediction of the robot's behaviour in numerous situations before expanding into a more integrated control system.

Given that the aim is to develop a robot control system that can navigate in an unknown environment, the collision avoidance algorithm will form the basis of this controller. A wall following algorithm is introduced later in the chapter, and the controller is then enhanced by the addition of an acceleration filter and appropriate sensor horizons. However, a completed model of an engineering control system demands an investigation into the system's interaction with its environment. To this effect, this chapter will conclude by incorporating the dynamics of the free-flyer relative to a rotating co-ordinate frame, namely, the orbiting space station.

3.2 Collision Avoidance

To ensure robot safety due to impacts on the walls and surroundings, robust collision avoidance and speed control must be implemented. This provides the basis for the initial, simple architecture before further behaviours are added and a more integrated system is established.

For the collision avoidance mechanism to be implemented, the Braitenberg philosophy (*Braitenberg, 1984*) of directly connecting sensor output with corresponding motion is adopted, as discussed in **chapter 1**. Explained in **chapter 2**, the robot's virtual hardware contains six proximity sensors, placed in a regular configuration with two on each of three orthogonal planes, as detailed in **figure 3.1**. These three planes form the reference frame of the robot: x - y, y - z, z - x.

The proximity sensors directly measure the distance to any obstacle that lies along the sensor line-of-sight and within its horizon, d, which is recorded as a scalar, with the direction determined by adding or subtracting the scalar value. This therefore allows a method for collision avoidance by providing a simple estimation of the environment.

The sensory information gathered becomes a direct input to the control system. The control system is driven by the sensor outputs from each pair of detectors, and directly calculates the acceleration needed to avoid colliding with the obstacle. This is done by an inverse relationship between the information from each sensor and the distance to the obstacle to give a resultant signal strength S_i for each sensor i.

$$s_i = \frac{1}{d_i} \qquad i = 1\text{-}6 \tag{3.1}$$

where d_i is the scalar distance recorded by sensor i.

The two sensor signals on each plane are then combined in a Braitenberg fashion (*Braitenberg, 1984*) whereby we introduce coefficients that vary the weighting of each sensor. Then, the resultant acceleration required to avoid colliding with an obstacle can be calculated as:

$$\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = \begin{bmatrix} c_1 & c_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & c_3 & c_4 & 0 & 0 \\ 0 & 0 & 0 & 0 & c_5 & c_6 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \end{bmatrix} \tag{3.2}$$

where a_x, a_y and a_z are the collision avoidance accelerations desired on each axis and c_i are the Braitenberg coefficients. If there were no non-zero coefficients, there would be an influence of obstacles sensed in all lines of sight on each component of acceleration. However, it is more practical for acceleration components to be only influenced by obstacles detected in the same direction (positive or negative). This may be summarised as:

$$\underline{a}_{av} = \underline{M} \cdot \underline{S} \tag{3.3}$$

where matrix M is a matrix representation of the behaviour and can be controlled by altering the values of the Braitenberg coefficients, $\underline{S}$ is the vector of sensor outputs and $\underline{a}_{av}$ is the required collision avoidance acceleration commanded.

The above equations detail how the acceleration is calculated from the sensor information. Since the signal from each sensor is inversely proportional to the distance to the obstacle it encounters, the signal increases as the robot approaches a collision hazard. The desired acceleration in this plane is then calculated proportional to the sensor signal (where the Braitenberg coefficients have the affect of altering the strength of the behaviour and will be demonstrated to do so later in this chapter).

Shown in **figure 3.2**, the acceleration calculated to ensure a collision free path is filtered to provide an on-off signal to the robot's thrusters. In this manner the

thrusters remain off until the threshold acceleration, a_t, is reached, at which point the thrusters are switched on at a fixed level q for fixed time period, Δt.

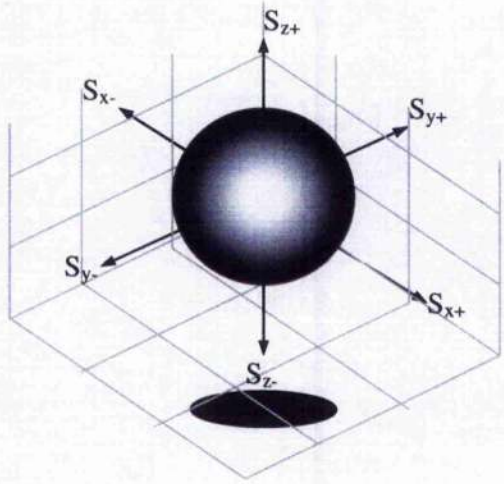
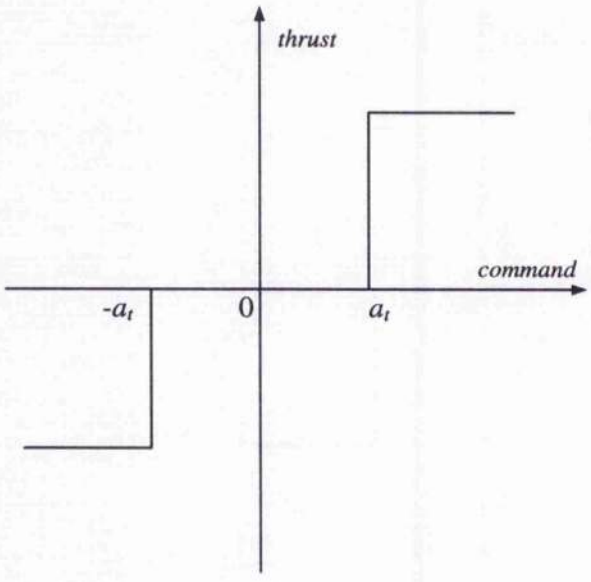


Figure 3.1: Robot model where sensor lines of sight are indicated by arrows.



If $a_i > a_t$ then
thrust = q.

If $a_i > -a_t$ then
thrust = $-q$.

Else,
thrust = 0.

Where a_i is the collision avoidance acceleration. a_t is the threshold acceleration, and q is the thrust provided. (-ve thrust indicates thrust acting in the –ve opposite direction, i.e. along the –ve axis.

Figure 3.2: When acceleration threshold is reached, thrusters are pulsed on.

Given the thruster fire time-step, Δt, the final Δv for each plane is then calculated:

$$\Delta v_i = a_{av_i} \Delta t \tag{3.4}$$

where i is defined as 1-3.

Therefore taking this expression, the new velocity in each plane is then calculated as:

$$v_i = v_{i_{prev}} + \Delta v_i \tag{3.5}$$

This method of integrating the ordinary differential equations (ODE) of motion is known as an Euler integration, taken from:

$$y_{n+1} = y_n + h\,\mathrm{f}\left(x_n, y_n\right) \tag{3.5b}$$

which advances the solution from y_n to $y_{n+1} \equiv y_n + h$, with step h.

The formula is unsymmetrical and advances the solution through an interval h, but uses derivative information only at the beginning of that interval. This means that the step's error is only one power of h smaller than the correction, labelling it first-order accuracy. A more accurate method would make the use of a step like (3.5b) to take a 'trial' step to the midpoint of the interval. It would then use the value of both x and y at that midpoint to compute the real step across the whole interval, thus making the method second order, known as the *Runge-Kutta* method (*Press, Teukolsky, Vetterling, Flannery, 2002*). However, the Euler method, though the least accurate method for integrating an ODE compared with other methods running at the same step-size, is the simplest, and, so long as the step-size is kept small, is sufficient for the purpose of this investigation.

This new velocity, obtained from equation 3.5, is then used to calculate the new position of the robot as:

$$x = x_{prev} + v_1 \Delta t \tag{3.6a}$$

$$y = y_{prev} + v_2 \Delta t \tag{3.6b}$$

$$z = z_{prev} + v_3 \Delta t \tag{3.6c}$$

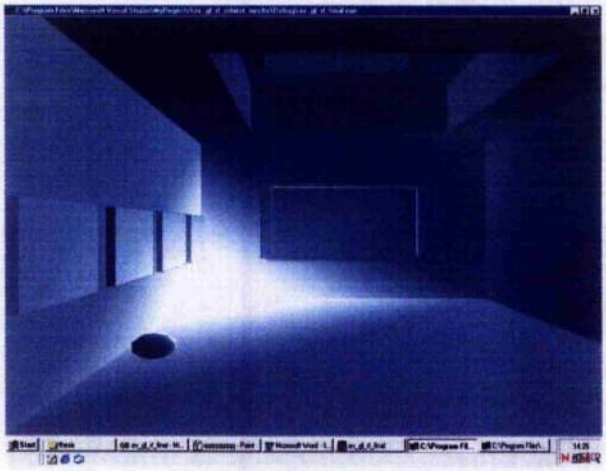
At each time-step the new position of the robot is calculated and stored in the virtual environment by looping though this portion of pseudo-code embedded in the simulation.

3.3 Initial Testing

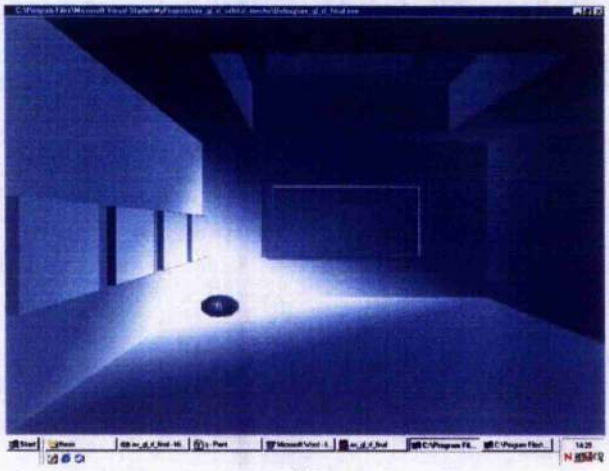
Figure 3.3 shows a series of still frames capturing the motion of the free-flyer manoeuvring in its virtual environment (described in **Chapter 2**) during a run of the computer simulation where the matrix of Braitenberg constants for this initial run is:

$$\mathbf{M} = \begin{bmatrix} 0.5 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & 0.5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0.5 \end{bmatrix}$$

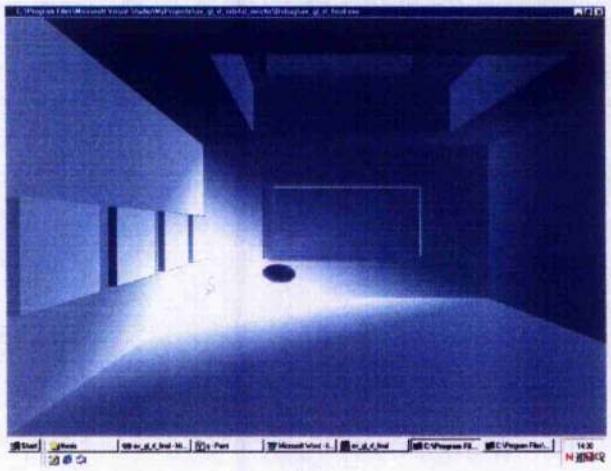
The snapshots capture the obstacle avoidance algorithm over a period of 100 seconds and aid in the perception of the free-flyer's motion and interaction with its three-dimensional environment, demonstrating collision avoidance at the top right-hand corner of the workspace. In **figure 3.4**, the trajectory of the free-flyer's motion whilst exhibiting the obstacle avoidance behaviour is shown on a simplified three-dimensional grid for easy observation, with a start location (3, 3, 3) from the origin of the workspace. This grid represents the skeleton of the virtual environment created using the Opengl® interface described in **chapter 2** and again shown in **figure 3.3**. This allows the robot's motion with respect to the environment walls to be displayed. **Figure 3.5** plots the x, y and z components of this test trace over a period of 200 seconds, demonstrating the collision avoidance behaviour along each axis. Given that the x, y and z components of the enclosed environment are 15 metres in length, the plots indicate the success of the collision avoidance behaviour by demonstrating that the robot at no point reaches a distance of 15 metres in any direction and so avoids colliding with the environment boundaries.



(a)



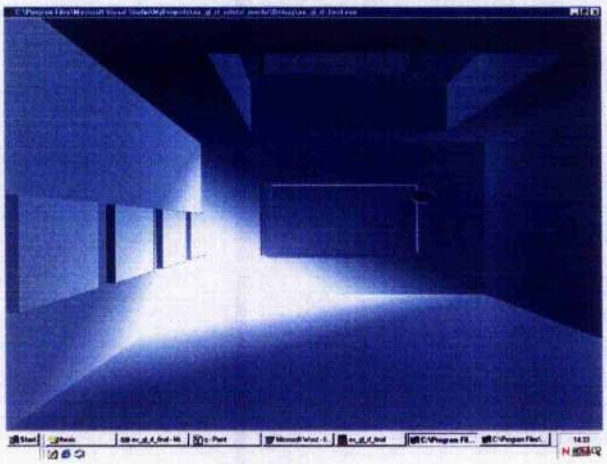
(b)



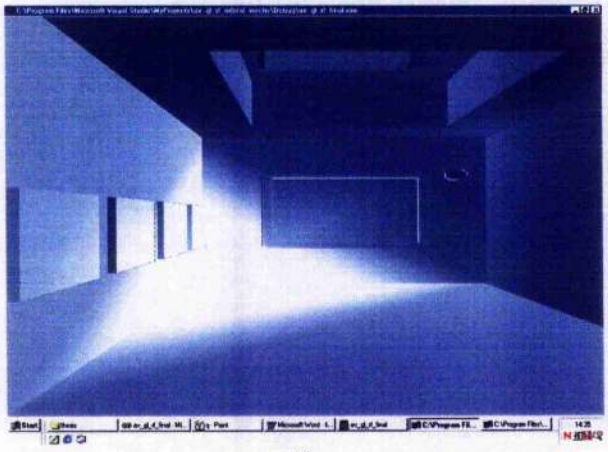
(c)

Figure 3.3(a) - (i): Frames capture the motion of robot.

(d)

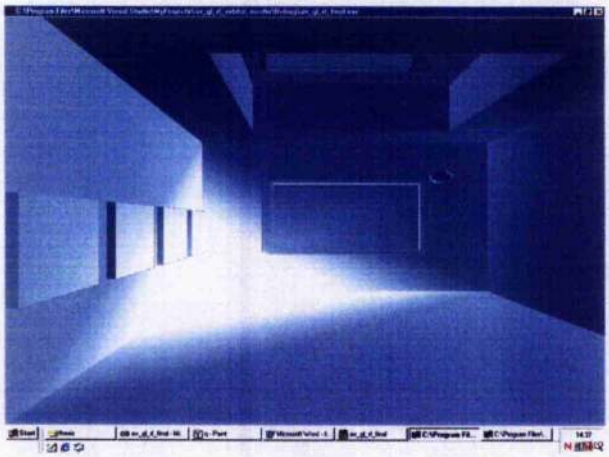


(e)

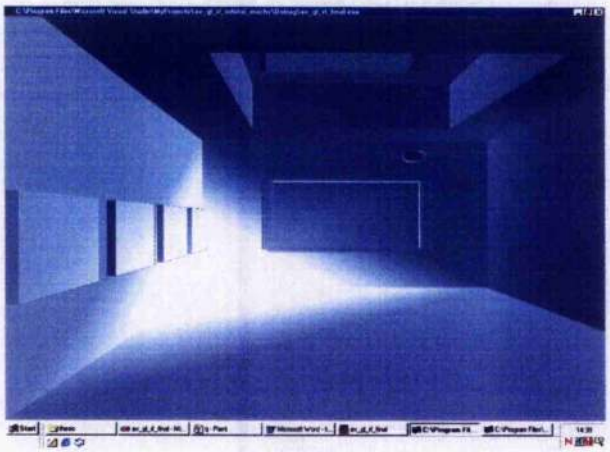


(f)

Figure 3.3(a) - (i): Frames capture the motion of robot.



(g)



(h)

(i)

Figure 3.3(a) - (i): Frames capture the motion of robot.

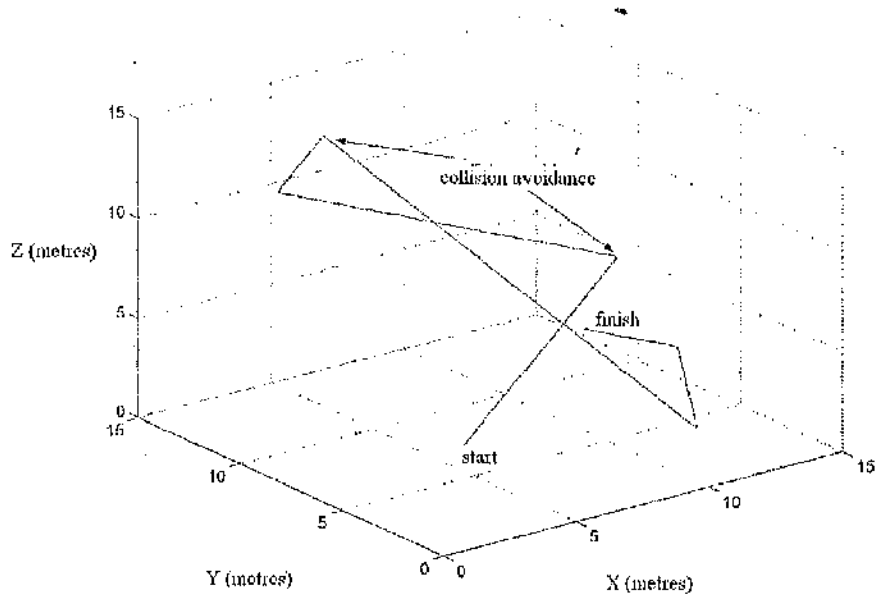


Figure 3.4: Trace of vehicle trajectory exhibiting collision avoidance behaviour for a period of 200 seconds.

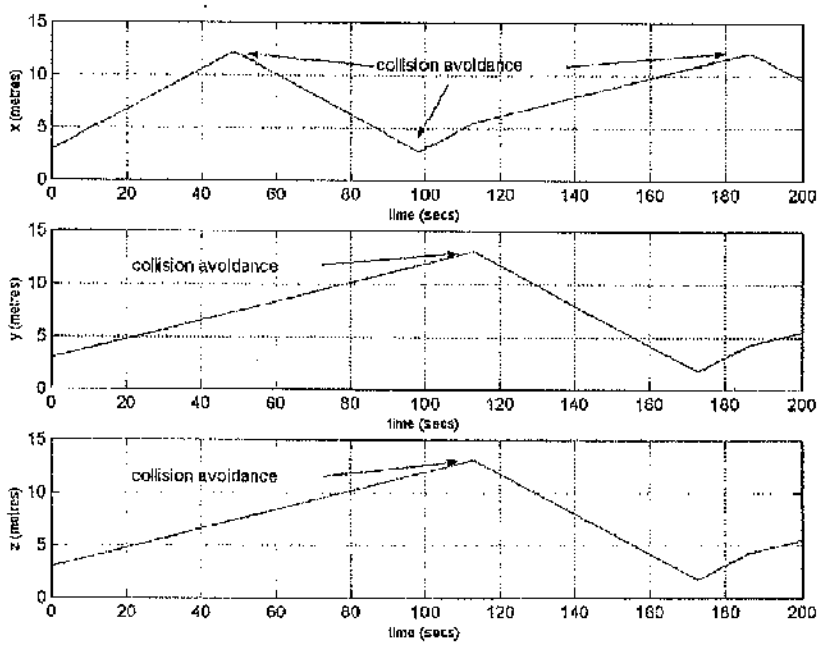


Figure 3.5: The vehicle trajectory showing collision avoidance.

3.4 Effect of the Braitenberg Coefficients

The Braitenberg coefficients, as mentioned earlier, add an element of versatility to the collision avoidance behaviour by influencing the distance at which the robot recoils from a wall, and is shown in Fig 3.5b

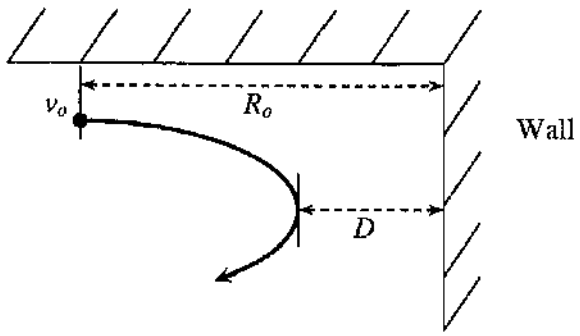


Figure 3.5b: Figure showing robot recoil.

where:

v_o = speed, v, at distance R_o from wall

R_o = sensor horizon

D = minimum distance, d, to wall

given that:

$$v = 0 \text{ at } d = D \text{ and } v = \dot{d}$$

and in this instance, signal strength S is calculated to be inversely proportional to distance from the wall, such that S=1/d.

Then from **equation (3.3)**, it can be shown that, for the 1-D case:

$$a_{av} = c \cdot S \tag{3.7}$$

Substituting the expression for signal strength then gives:

$$\ddot{d} = a_{av} = c \cdot \frac{1}{d} \tag{3.8}$$

and if $v = \dot{d}$, then:

$$\ddot{d} = \dot{v} = \frac{\partial v}{\partial d} \cdot \frac{\partial d}{\partial t} \tag{3.9}$$

Rearranging (3.9) gives:

$$\ddot{d} = \dot{v} = \frac{\partial d}{\partial t} \cdot \frac{\partial v}{\partial d} = v \cdot \frac{\partial v}{\partial d} \tag{3.9b}$$

therefore, substituting **(3.8)** into **(3.9)** gives:

$$v \cdot \frac{\partial v}{\partial d} = c \cdot \frac{1}{d} \tag{3.10}$$

which can be re-arranged to give:

$$v \cdot \partial v = c \cdot \frac{\partial d}{d} \tag{3.11}$$

and then integrating the above gives:

$$\frac{1}{2}(v^2 - v_o^2) = c(\log(d) - \log(d_o)) \tag{3.12}$$

therefore since $v = 0$ at $d = D$,

$$\frac{1}{2}(v_o^2) = c(\log(D) - \log(d_o)) \tag{3.13}$$

which this can be expressed as:

$$\frac{1}{2}(v_o^2) = -c(\log(\frac{D}{d_o})) \tag{3.14}$$

Or, alternatively,

$$\frac{1}{2}(v_o^2) = c(\log(\frac{d_o}{D})) \tag{3.14b}$$

This expression can be re-arranged to give expressions for the Braitenberg coefficient and the minimum approach distance to the wall as:

$$c = \frac{v_o^2}{2\log(d_o/D)} \tag{3.15}$$

Where it may be seen that c is dependent on v_o^2. It is assumed that, due to safety constraints on the free-flyer, there is a maximum permittable value of v_o (taken to be 0.1m/s) which thus causes a restraint on the Braitenberg values and thus the behaviour of the robot.

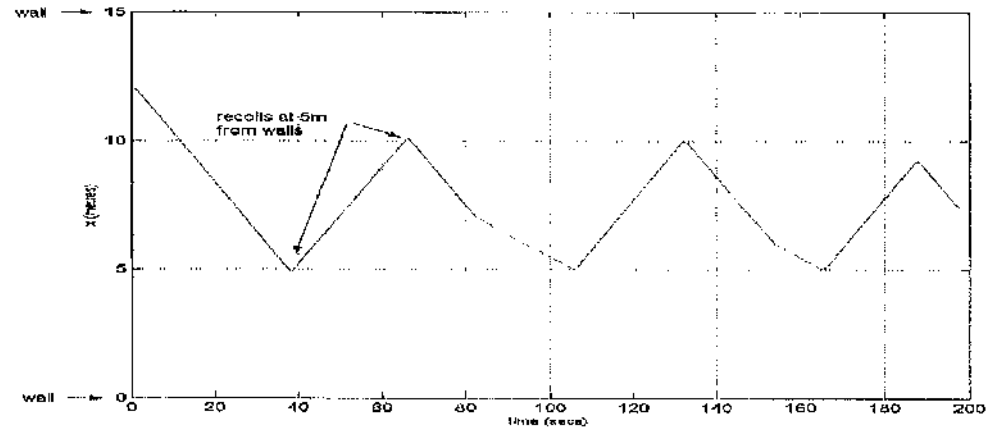
Equation (3.15) may be re-arranged to give:

$$D = d_o \exp(-v_o^2/2c) \tag{3.16}$$

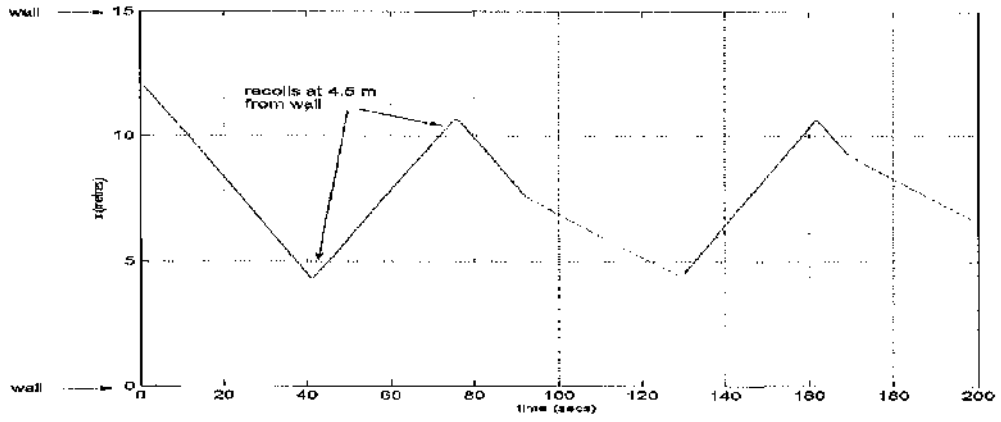
From (3.16), it is seen that as c decreases, D decreases and as v_o increases, D decreases. This scaling law can be used for coefficient selection as is shown in the following plots.

Figure 3.8(a) through **(f)** plots the robot's trajectory exhibiting the collision avoidance behaviour with progressively decreasing Braitenberg coefficients, c_i $(i = 1-6)$. As is demonstrated, by decreasing the coefficient, the robot delays recoil until it comes closer to the wall. In **figure 3.8(a)** where the Braitenberg coefficient is 1, the robot recoils from the wall at a distance of approximately 5 metres. Progressing to **figure 3.8(e)**, where the Braitenberg coefficient has been reduced to 0.1, the robot recoils at 1 metre from the wall. In a concept introduced by Valentino Braitenberg in his book 'Vehicles' (*Braitenberg, 1984*), he suggests that such differing relationships between a vehicle and its source (its stimulus) may give the appearance outwardly of a vehicle exhibiting behavioural characteristics. This may be adapted here by suggesting that a reduction in the Braitenberg coefficient could be seen to increase the

robot's 'boldness', as the robot veers closer to a potential collision. Reducing the Braitenberg coefficient too far can result in dangerous behaviour with the possibility of impact, which is shown to occur in **figure 3.8(f)** where the Braitenberg coefficient has been reduced to 0.15. However, having a large coefficient reduces the space available to the robot, and results in relatively large accumulated Δv: there are 6 instances of collision avoidance accelerations where $c_i = 1$ but only 2 in the case where $c_i = 0.2$. In the context of developing a free-flying space robot, however, a coefficient is chosen which will balance safety (v_a restricted) with efficiency. To this end, a reasonable coefficient value of 0.3 is chosen which, from **figure 3.8(d)**, causes the robot to recoil at approximately 2 metres from the walls.

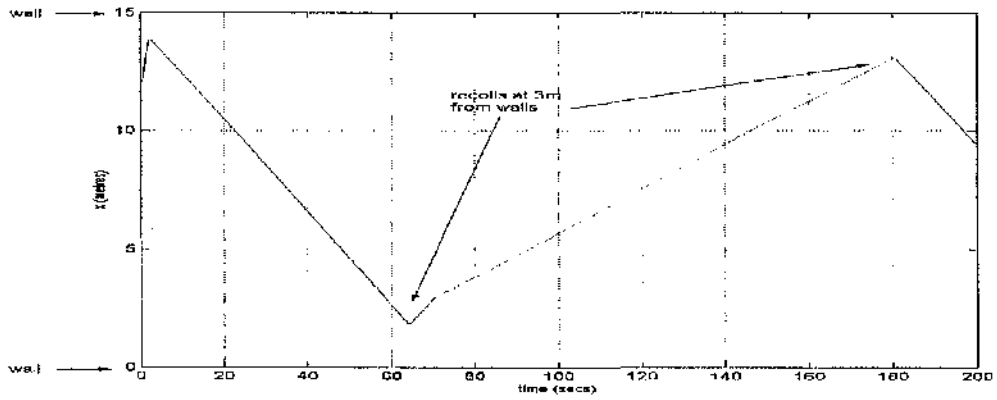


(a) $c_i = 1$

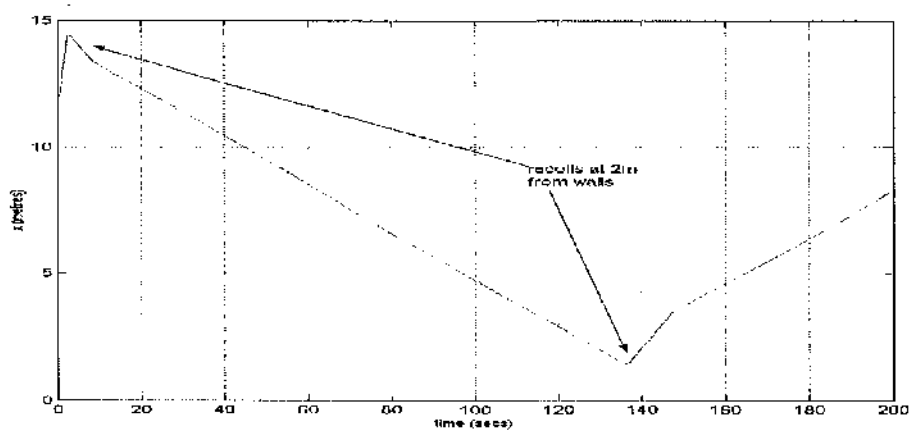


(b) $c_i = 0.7$

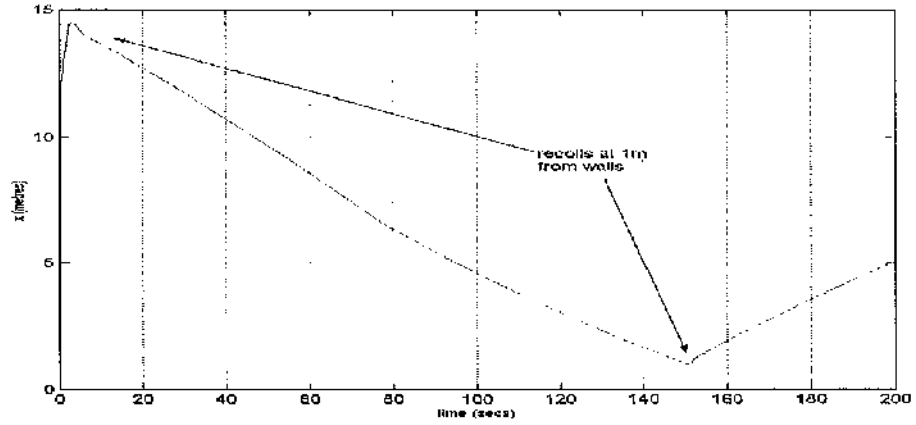
Figures 3.6(a) - (f): The effect of altering the Braitenberg coefficient c_i on collision avoidance efficiency.



(c) $c_i = 0.5$

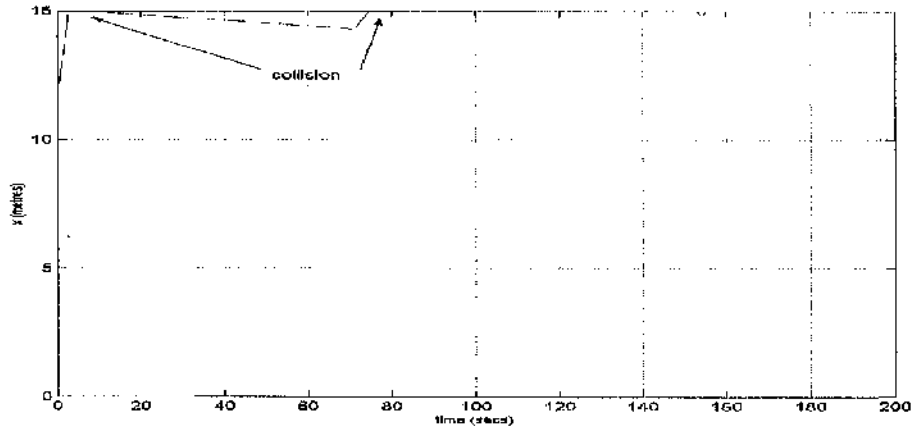


(d) $c_i = 0.3$



(e) $c_i = 0.2$

Figures 3.6(a) - (f): The effect of altering the Braitenberg coefficient c_i on collision avoidance efficiency.



(f) $c_i = 0.15$

Figures 3.6(a) - (f): The effect of altering the Braitenberg coefficient c_i on collision avoidance efficiency.

3.5 Wall Following

Currently the methodology, although successful in its aim, is basic and with limited practical use. The robot merely manoeuvres randomly around in a simulated environment avoiding collision hazards. So to this end, expansion of the model begins by introducing a wall-following algorithm. The benefit of this algorithm is to aid the robot in finding its way more easily in a constrained environment by following wall surfaces encountered. Information gathered from the robot sensors detecting a wall surface again drives the controller.

The control system is driven by the sensor outputs from each plane of detectors, and directly calculates the acceleration needed to follow the wall encountered. This is done by a proportional relationship between the information from each sensor and the distance to the wall to give a resultant signal strength for each sensor. The two sensor signals on each plane are again combined in a Braitenberg fashion (*Braitenberg, 1984*) to calculate the resultant acceleration required to follow the wall surface.

As with the avoidance algorithm, the acceleration required for the wall following algorithm is calculated as:

$$\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = \begin{bmatrix} c_1 & c_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & c_3 & c_4 & 0 & 0 \\ 0 & 0 & 0 & 0 & c_5 & c_6 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \end{bmatrix} \tag{3.17}$$

where a_x, a_y and a_z are the wall following accelerations desired on each axis and c_i are the Braitenberg coefficients as before. This may be summarised as:

$$\underline{a}_{wf} = \underline{M} \cdot \underline{S} \tag{3.18}$$

where **M** is a matrix of Braitenberg coefficients controlling the strength of this behaviour. It may be noted that the coefficients in the wall following matrix will be of opposite sign to those in the collision avoidance matrix (the values of c_i become negative) to entertain a behaviour that attracts to the walls rather than recoils from them.

In this instance, signal strength S_i is calculated to be proportional to the measured distance, such that:

$$S_i = d_i \quad i=1\text{-}6 \tag{3.19}$$

where d_i is the distance recorded by sensor i.

Therefore, the signal decreases as the robot approaches a wall and increases as the robot departs from the wall. The acceleration in this plane is then calculated proportional to the sensor signal but it is in the opposite direction to the obstacle avoidance acceleration. As with the avoidance algorithm, the acceleration obtained from the wall following algorithm is re-calculated at each new time-step and input to

the control system. The robot oscillates along the wall surfaces due to the influence of the opposing attractive and repulsive forces acting on it.

Within the control system, the components of each behaviour (wall following and obstacle avoidance) are weighted and summed for each plane to give a final resultant acceleration:

$$\underline{a}_{total} = \lambda_{avoid}\underline{a}_{av} + \lambda_{wall\_follow}\underline{a}_{wf} \tag{3.20}$$

Figure 3.7 demonstrates the path taken by the robot during an animated test run of the model where the matrices of Braitenberg constants for this initial run are:

$$\underline{M}_{av} = \begin{bmatrix} 0.5 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & 0.5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0.5 \end{bmatrix} \tag{3.21}$$

$$\underline{M}_{wf} = \begin{bmatrix} -0.5 & -0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & -0.5 & -0.5 & 0 & 0 \\ 0 & 0 & 0 & 0 & -0.5 & -0.5 \end{bmatrix} \tag{3.22}$$

The run was performed over a period of 500 seconds with the robot exhibiting both the wall following and collision avoidance algorithms combined. **Figure 3.8** displays the x, y and z components of the vehicle's trajectory separately to ease interpretation.

As is evident from both **figures 3.7** and **3.8**, the robot spends the majority of its time close to, and oscillating along wall surfaces (wall surfaces are positioned at 0 metres and 15 metres along each axis). This can be compared further with **figures 3.4** and **3.5** where very little time is spent close to wall surfaces. The oscillation occurs due to the combining effect of the attractive and repulsive forces from the two algorithms, and again altering the weighting of each algorithm alters the amplitude and frequency of the oscillation – shown in **figure 3.7**.

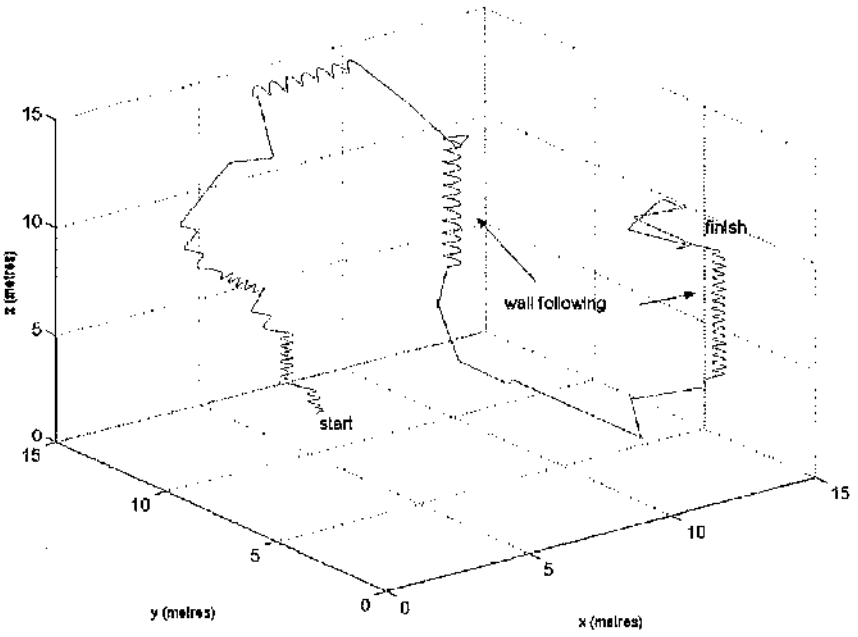


Figure 3.7: Trajectory of the robot influenced by both wall following and avoidance algorithms.

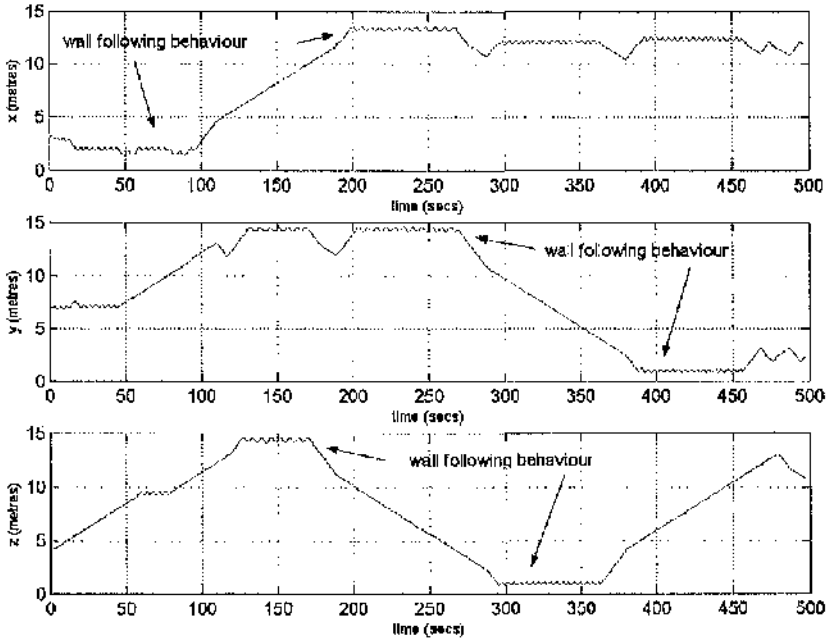


Figure 3.8: x, y & z components of vehicle trajectory shown in **figure 3.7**.

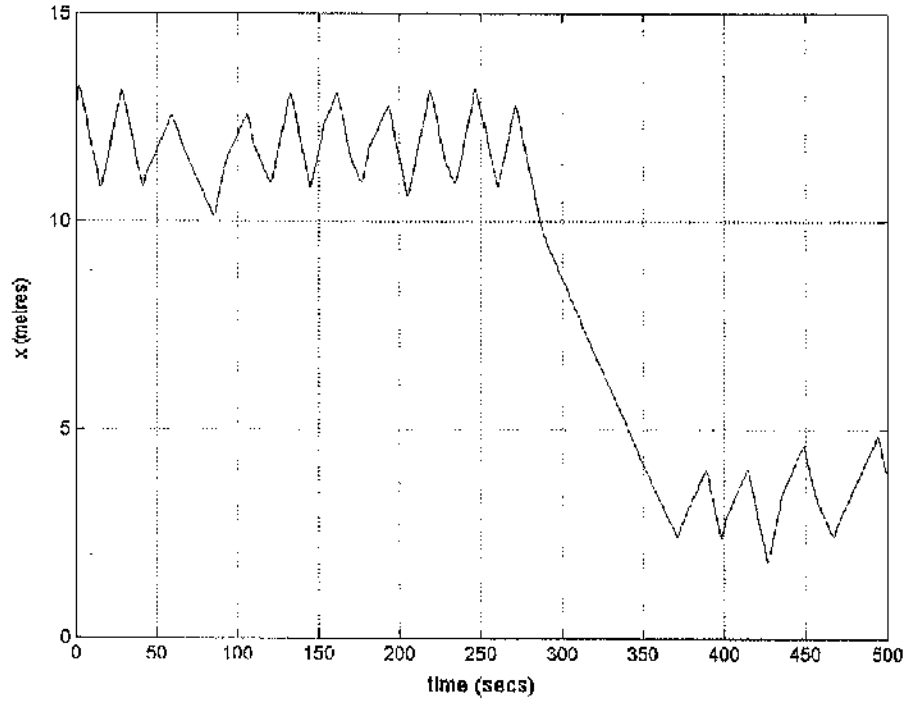
The global behaviour of the robot may be changed by altering the weighting values, λ, on each individual behaviour. In **figure 3.9(a) - (d)** the motion of the robot along the x-axis is shown over a period of 500 seconds whereby the ratio of the obstacle avoidance weighting to the wall following weighting is reduced from 10:1 to 1:1. In **figure 3.9(a)**, where the ratio of weightings is 10:1, the robot's wall following behaviour is inefficient; the oscillations along wall surfaces have a large amplitude. Progressing through to **figure 3.9(d)**, where the ratio is 1:1, the wall following motion becomes smoother reducing the accumulated Δv and hence increasing the robot's efficiency. It should be noted that the difference in the macroscopic behaviour of each of the **figures 3.9(a)** to **(d)** can be attributed to the different paths generated as a direct result of the difference in the weighting ratio between the avoidance and wall following behaviours, as this difference will affect the robot's position in space, and so being influenced by the environment, will affect the robot's trajectory. In this simple design these weightings are constant, however, later as the system becomes more advanced, these weightings are replaced by variable weightings that are dependant on vehicle state.

3.6 Effect of Altering Acceleration Threshold

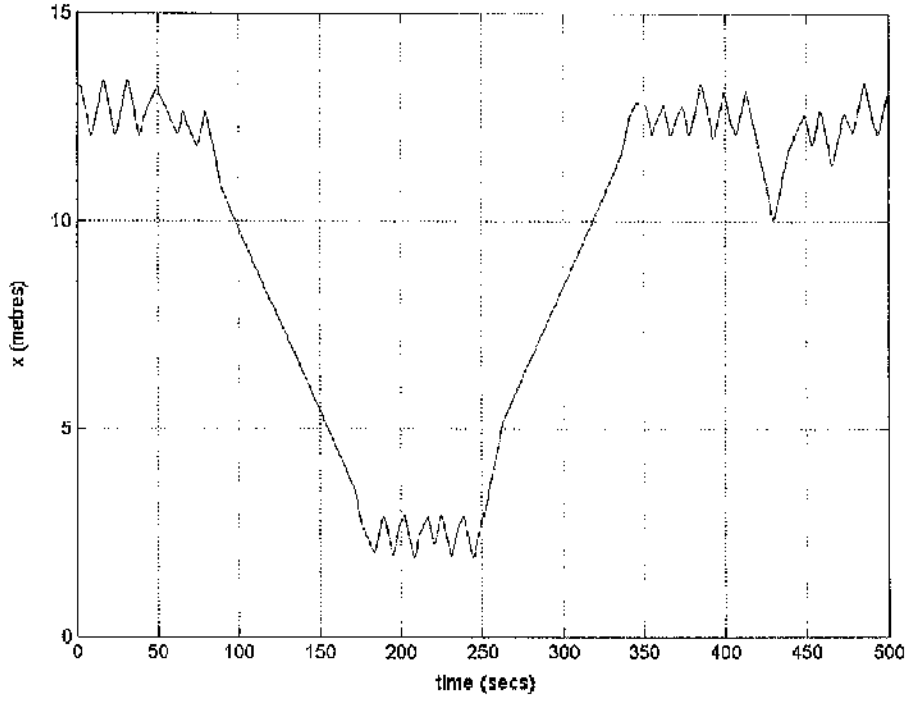
As was mentioned in **section 3.2**, the robot thrusters are pulsed on when a threshold value of desired acceleration, a_t, needed to execute the manoeuvre is reached. When this value is reached, thrusters are pulsed on producing acceleration q . The effect of altering a_t on the vehicle trajectory is interesting to observe and is shown in **figures 3.10(a)** though **(c)**. **Figure 3.10(a)** through **(c)** shows three examples showing position along the x-axis over a period of 200 seconds, with corresponding positive x-axis threshold acceleration. In the first example, **3.10(a)**, the threshold acceleration, a_t, is held at 0.05 ms^{-2} and thrusters are pulsed on to produce a corresponding acceleration of 0.05ms^{-2} when this threshold is reached. In **3.10(b)**, the threshold acceleration is 0.1ms^{-2} and corresponding thruster activity produces 0.1ms^{-2} acceleration. In **figure 3.10(c)**, a_t is filtered at 0.2ms^{-2} with thrusters producing 0.2ms^{-2}. Firstly, when acceleration is filtered at a higher level, it is noted that the thrusters need only stay on for shorter periods of time to carry out the desired effect.

The change in direction required of the robot is also executed much quicker in the examples with a higher threshold value noted by the steeper recoil angle. This fast and efficient execution is desired, so long as there is a control on the maximum vehicle speed. Maintaining the robot's velocity to within a safe limit will reduce the possibility of collision and reduce the severity of the impact if a collision does occur. This is of utmost importance within crewed space vehicles, both for the safety of walls and surroundings, the hardware, and for the safety of the crew itself. To this end, a fixed speed controller is incorporated into the design at this stage to ensure the robot never exceeds a velocity of $0.1ms^{-1}$. In such circumstance that the desired speed becomes greater than the limit of $0.1ms^{-1}$, the speed controller ensures the speed remains at that limit, as shown in the following pseudo-code:

```
void speed_control()
 {
    if (vi >= 0.1)
          {
          vi = 0.1;
          }
    if (vi <= -0.1)
         {
          vi = -0.1;
          }
    }
```

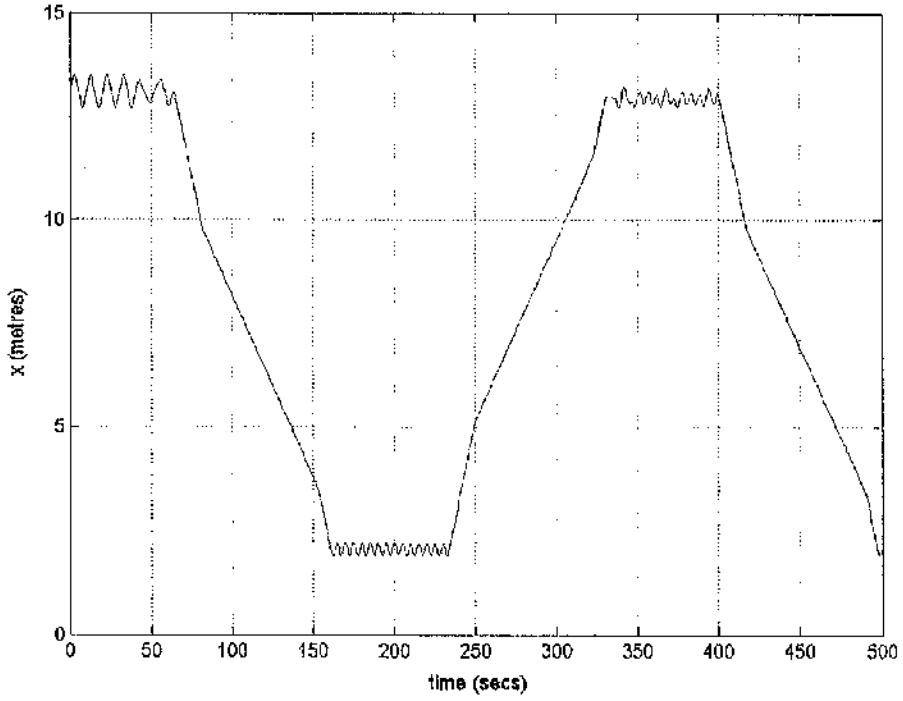


(a) weighting ratio ($\lambda_{av} : \lambda_{wf}$)= 10:1

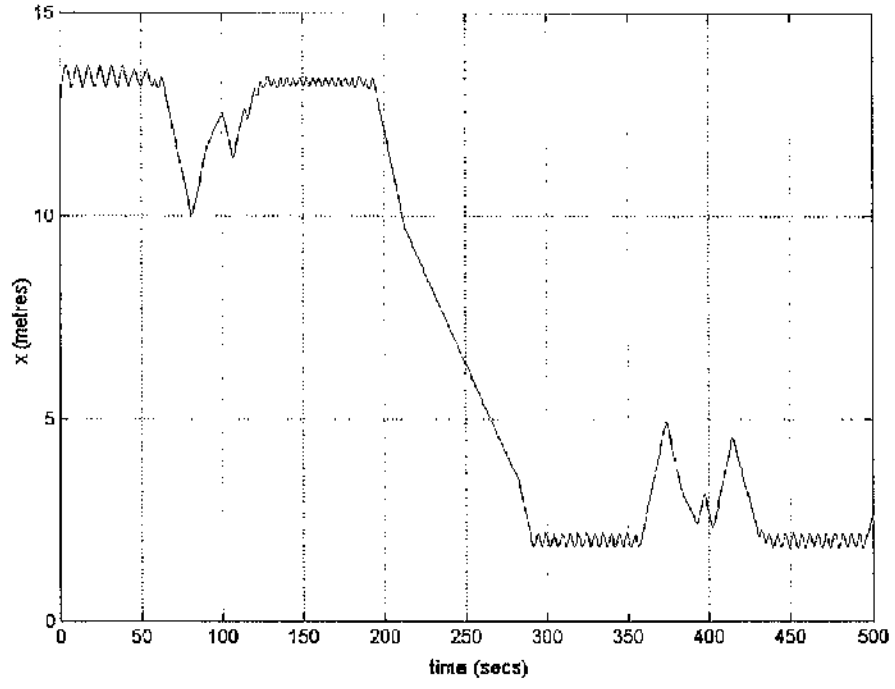


(b) weighting ratio ($\lambda_{av} : \lambda_{wf}$) = 5:1

Figures 3.9(a) - (d): The effect of altering the weighting ratio $\lambda_{av} : \lambda_{wf}$.

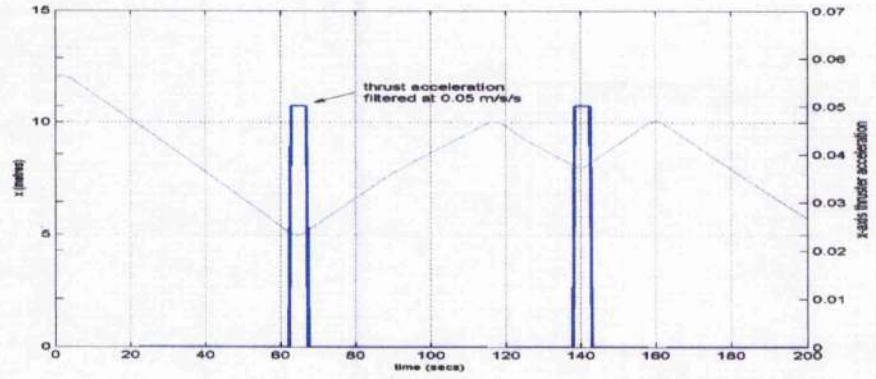


(c) weighting ratio ($\lambda_{av} : \lambda_{wf}$) = 2.5:1

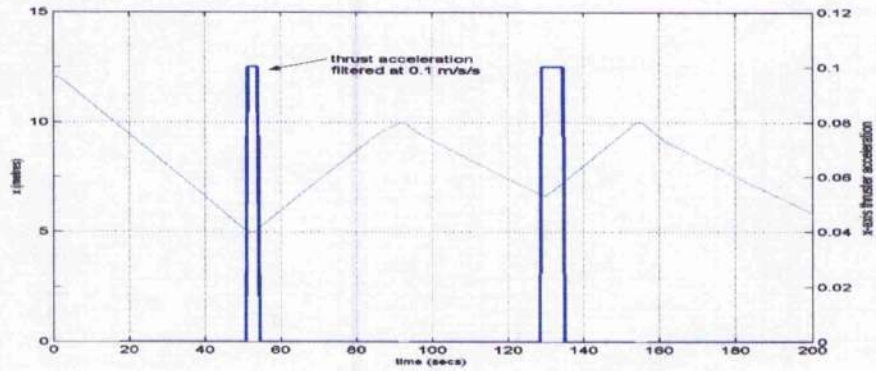


(d) weighting ratio ($\lambda_{av} : \lambda_{wf}$) = 1:1

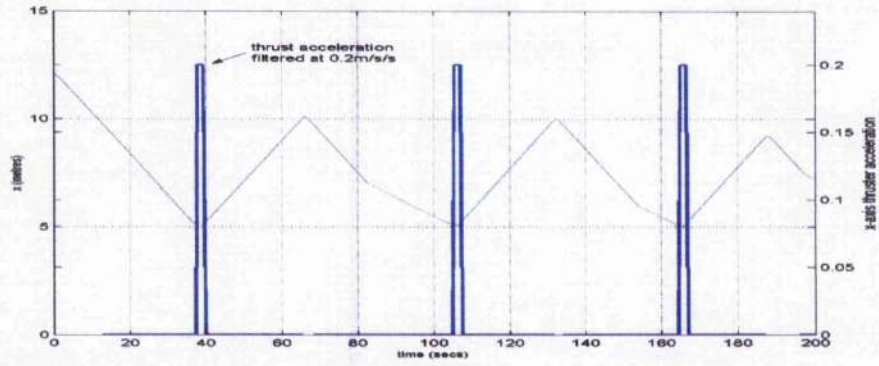
Figures 3.9(a) - (d): The effect of altering the weighting ratio $\lambda_{av} : \lambda_{wf}$.



(a) a_t at 0.05ms^{-2}



(b) a_t at 0.1ms^{-2}



(c) a_t at 0.2ms^{-2}

Figures 3.10(a) - (c): The effect of altering the threshold acceleration a_t.

3.7 Effect of Altering the Sensor Horizon

In the workspace of a free-flying robot, the safety of flight hardware can only be guaranteed though robust collision avoidance control where the spacecraft hardware is seen as a possible collision hazard, around which to navigate. The description of the method utilised here for collision avoidance employs the use of artificial forces as a function of the shortest distance between the robot and the obstacle. Collisions are prevented by making these forces *repulsive*, and 'wall following' is established by imparting *attractive* forces. Actuators producing forces proportionate to the sum of these specified forces control the motion of the free-flyer. However, therein lies the problem of determining the distance between the robot and its environment. Solving this requires either an accurate geometric model of the robot and its environment to be established a priori, or through gathering appropriate sensory information. The former is computationally expensive and so the latter becomes the viable choice here.

It is assumed that the robot is to be fitted with appropriate sensors, which are used to directly measure the distance to nearby obstacles. Since perfect measurements along every unit normal to every point on the surface of the robot is not possible, therein lies the problem of investigating and selecting the most appropriate sensor for the needs of this space application.

In chapter 2, the selection of proximity sensors was made from amongst the range of commercially available products and prototypes available, discussing the most appropriate category of sensing technology available for this space application whilst providing the desired coverage and sensor data quality. In this design, for simplicity purposes, the sensors are given a basic uniform horizon encapsulating the robot (as shown in **figure 3.11**). As mentioned before, there exists two sensors on each of the robot's three axes, and for this study, these sensors will only detect objects that lie along the sensor line-of-sight and within their horizon, R_x, R_y, R_z.

Figure 3.12(a) - (e) shows the effect reducing the sensor horizon has on the recoil distance from collision hazards. Progressing from **(a)** to **(e)**, the sensor horizon is reduced from 8 metres to 0.5 metres for the condition where the Braitenberg

coefficients $c_1 - c_6$ within matrix $\underline{M}$ are set at a low value of 0.3. When the sensor horizon is set to 8 metres, as in **(a)**, the robot recoils at a distance of approximately 2 metres from the hazard - similar to the case where there is unlimited sensor horizon shown in **figure 3.9(d)**.

Reducing the sensor horizon to 4 metres has no effect on reducing the recoil distance, nor does reducing to 2 metres, still exhibiting the same characteristics as **figure 3.9(d)**. This is due to having a low Braitenberg coefficient value which keeps the calculated acceleration too low to trigger thuster firing (also governed by the imposed restriction on v_o). However reducing to a 1-metre sensor horizon, as in **figure 3.12(d)**, the recoil distance is reduced to approximately 1 metre (thruster firing enabled, behaviour exhibited), and continuing to 0.5 metres the robot experiences collision with the environment (thrust activated too late to avoid collision).

In **figure 3.13(a) - (e)**, the effect of reducing the sensor horizon on the recoil distance is repeated for the condition where the Braitenberg coefficients are set at 0.6. In this instance, where the sensor horizon is set at both 8 metres and 4 metres, the robot recoils at a distance of approximately 3 metres from the hazard. Where the sensor horizon is reduced to 2 metres, the recoil distance is shown from **(c)** to be approximately 2 metres, then 1 metre at the 1-metre sensor horizon and again collision at 0.5 metres (higher Braitenberg coefficients activate thruster firing (a.k.a. behaviour response) earlier).

To show an example numerically, in terms of equation 3.16 viz:

$$D = d_o \exp(-v_o^2/2c) \tag{3.16}$$

Given that speed v_o is taken to be 0.1m/s (for safety constraints), at a distance d_o (sensor horizon) given as 2 metres from wall, then, if c is taken to be 0.3, this gives a stopping distance as:

$\underline{D = 1.9247 \text{ m}}$ as expected, shown in **fig 3.12(c)**

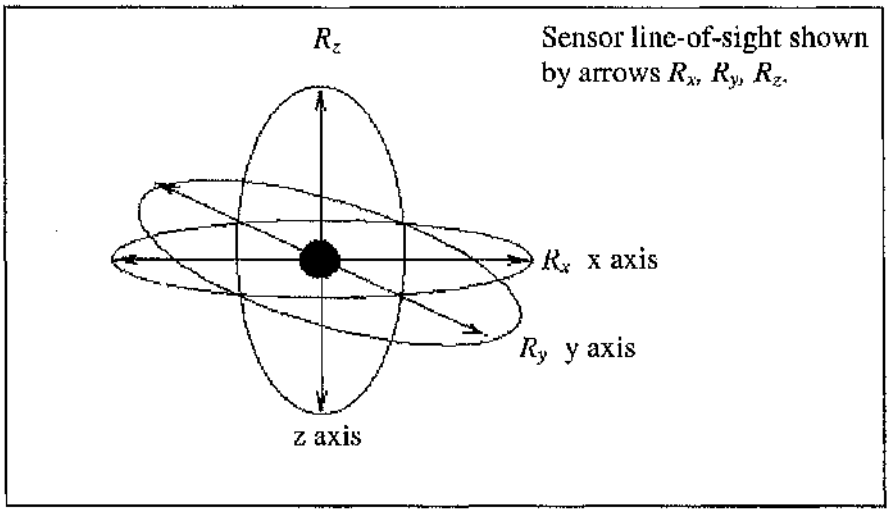
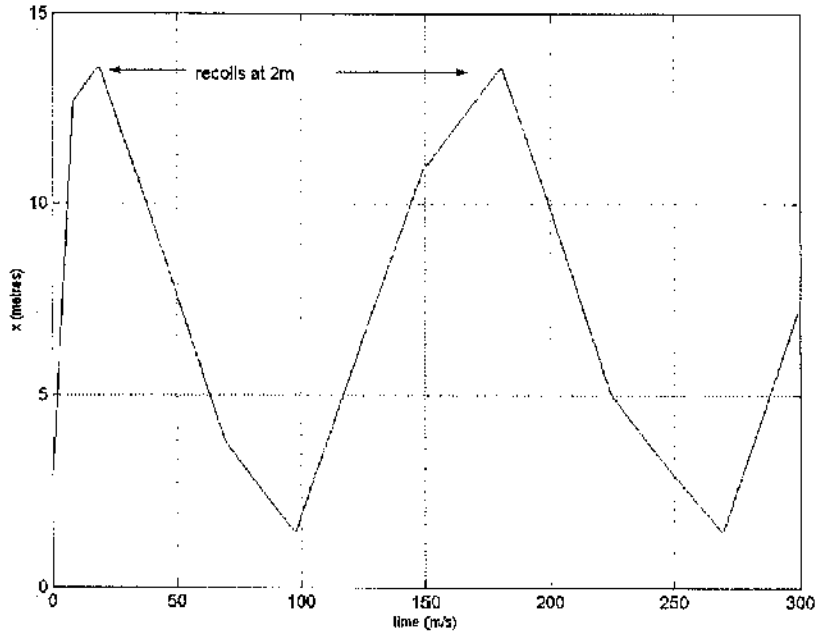
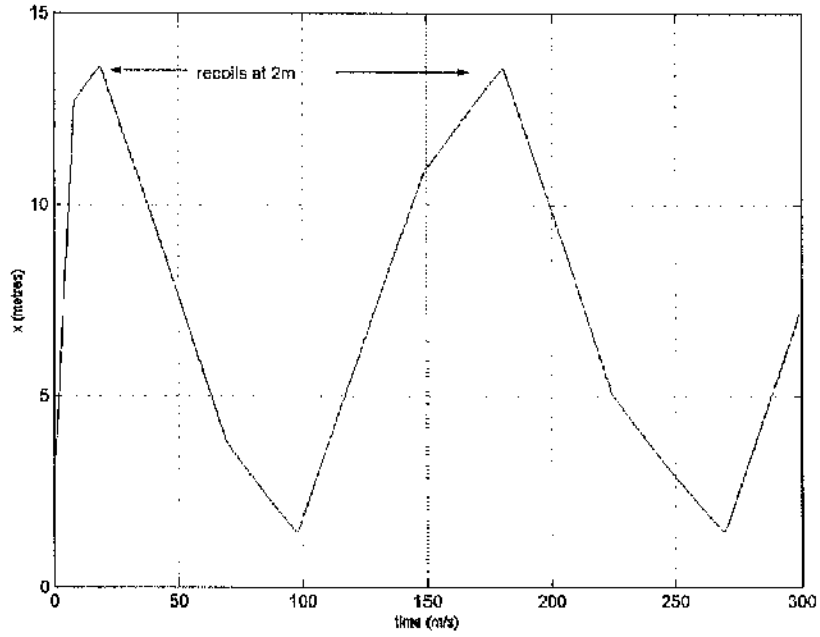


Figure 3.11: Sensor horizon and line-of-sight shown by enclosed arrows.

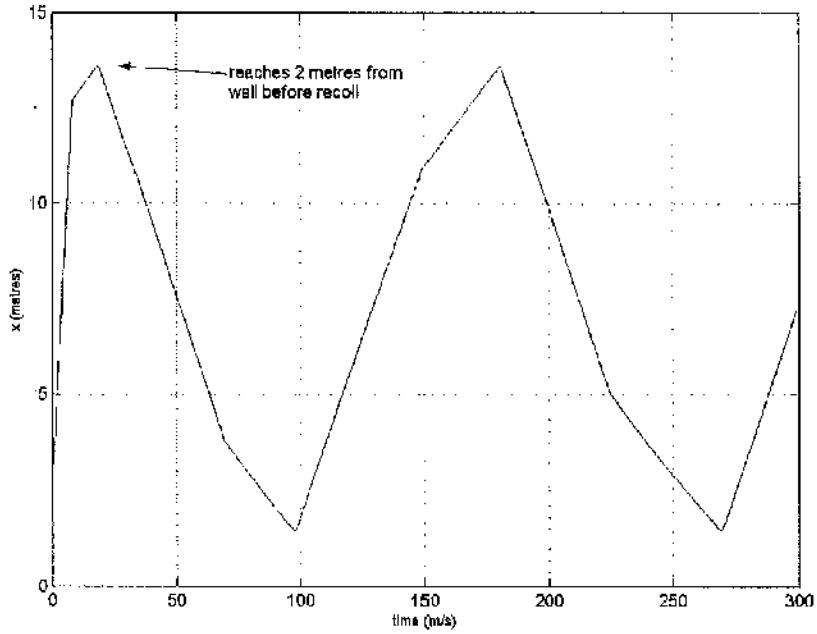


(a) Sensor horizon set at 8m

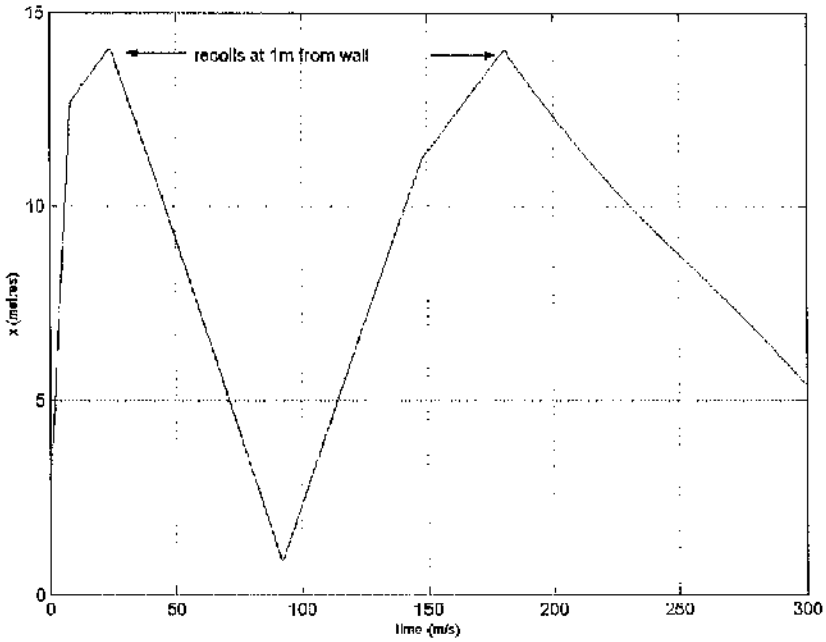


(b) Sensor horizon set at 4m

Figure 3.12(a) - (e): Altering the sensor horizon alters the recoil distance from the wall. $C_i = 0.3$ (i = 1-6).

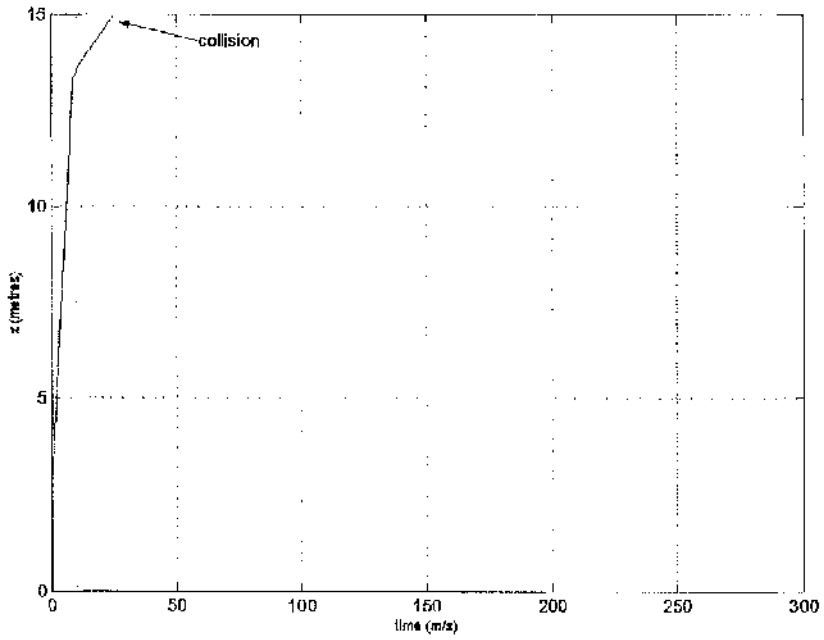


(c) Sensor horizon set at 2m



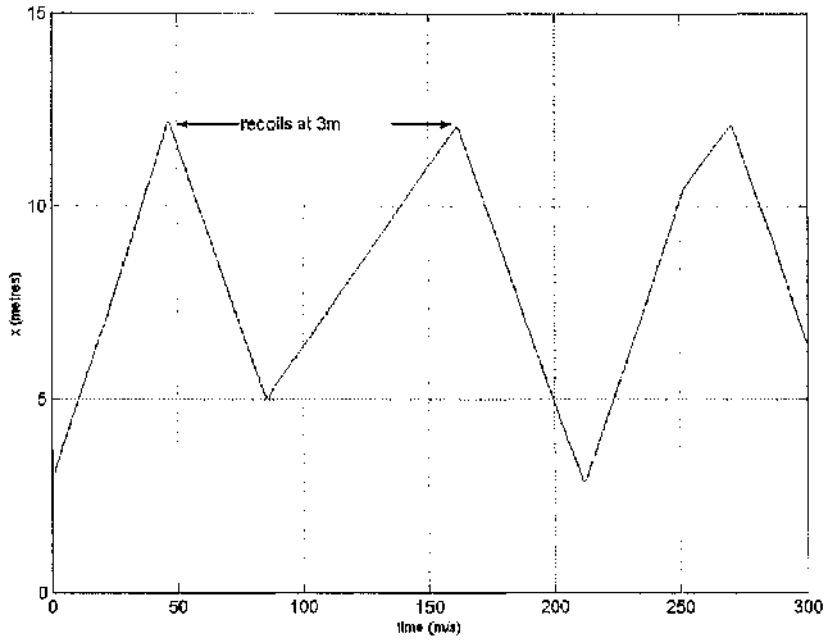
(d) Sensor horizon set at 1m

Figure 3.12(a) - (e): Altering the sensor horizon alters the recoil distance from the wall. $C_i = 0.3$ (i = 1-6).



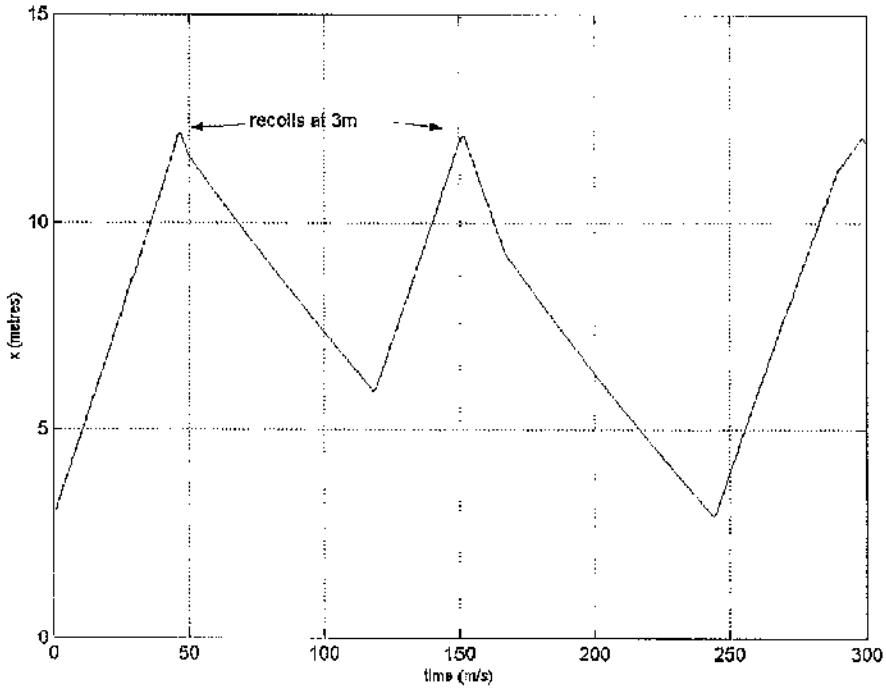
(e) Sensor horizon set at 0.5m

Figure 3.12(a) - (e): Altering the sensor horizon alters the recoil distance from the wall. $C_i = 0.3$ (i = 1-6).

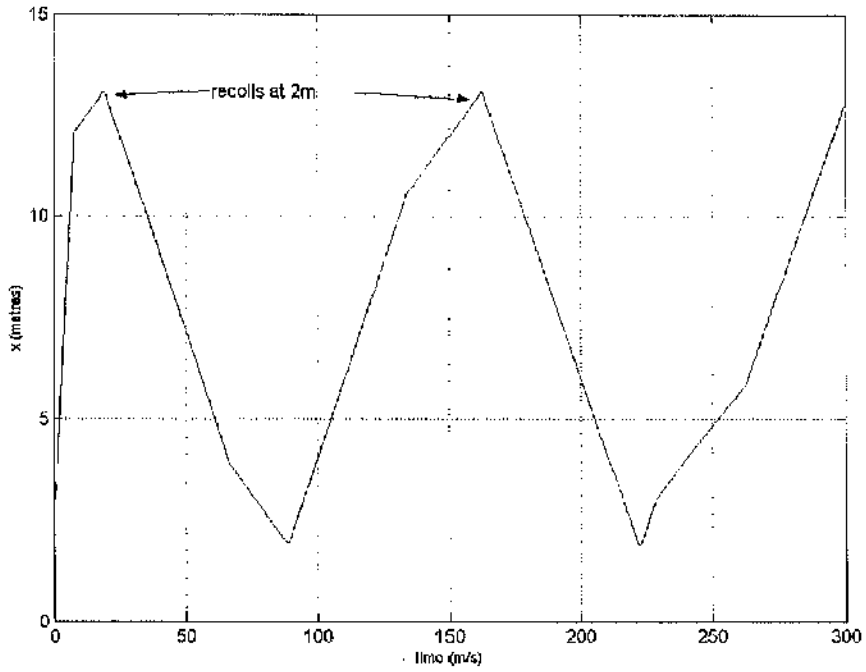


(a) Sensor horizon set at 8m

Figure 3.13(a) - (e): Altering the sensor horizon alters the recoil distance from the wall. $C_i = 0.6$ (i = 1-6).

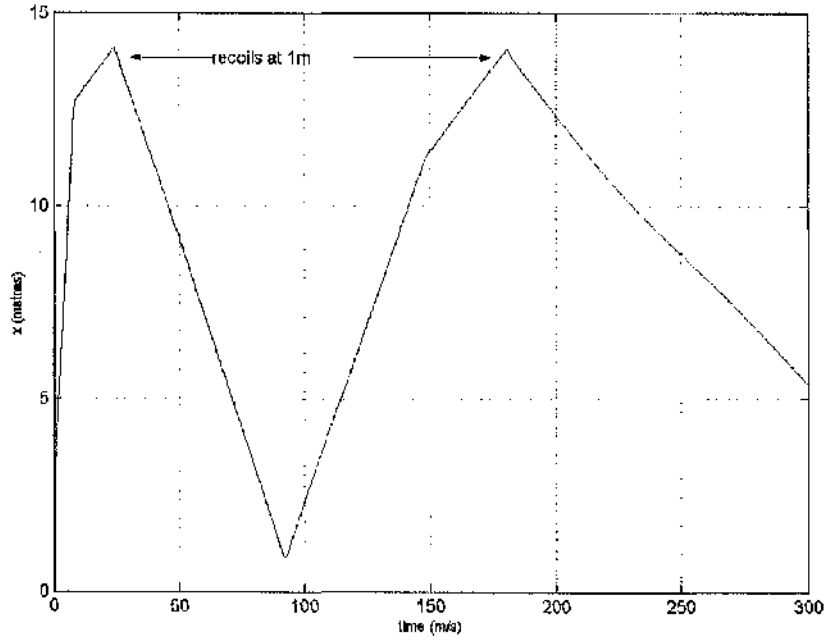


(b) Sensor horizon set at 4m

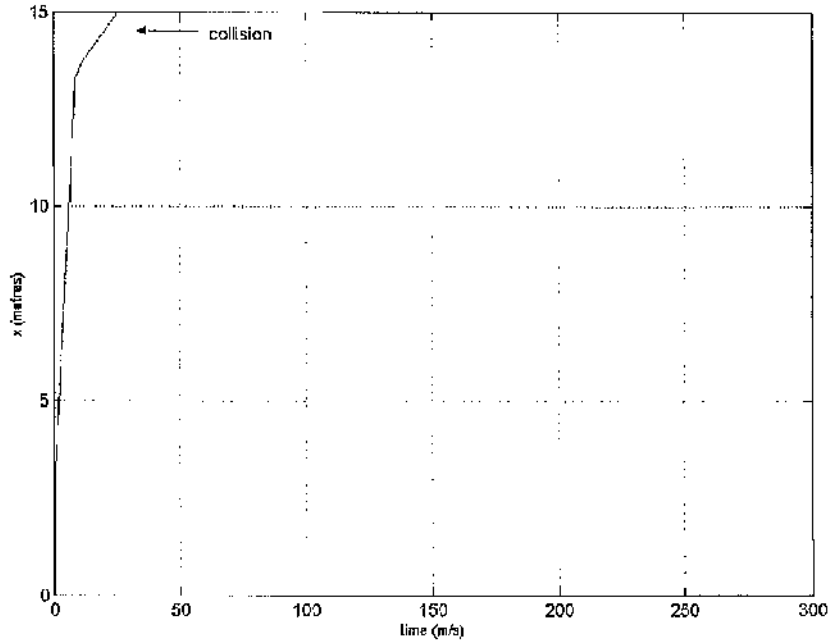


(c) Sensor horizon set at 2m

Figure 3.13(a) - (e): Altering the sensor horizon alters the recoil distance from the wall. $C_i = 0.6$ (i = 1-6).



(d) Sensor horizon set at 1m



(e) Sensor horizon set at 0.5m

Figure 3.13(a) - (e): Altering the sensor horizon alters the recoil distance from the wall. $C_i = 0.6$ (i = 1-6).

Further, the relationship:

$$v^2 = u^2 + 2ad \tag{3.24}$$

where v represents maximum velocity here chosen to be 0.1 ms^{-1}, u represents final velocity to be 0 ms^{-1}, a is acceleration chosen to be -0.2 ms^{-2} and d is stopping distance, may be re-written:

$$d = \frac{v^2 - u^2}{2a} \tag{3.24b}$$

which gives a maximum stopping distance of 0.025 m.

Increasing the Braitenberg coefficient introduces an increased level of versatility to the control system, allowing a greater variance in recoil distance, therefore encouraging a wider variance in exhibited behaviours (constrained by the imposed restriction on v_a). This is useful to allow the user to select the appropriate recoil distance required for the application, depending on safety requirements, maximum speed and acceleration.

Combining the choice of sensor made in **chapter 2,** which had a horizon of 1 metre; with versatile behaviour selection coefficients, and a desire to keep recoil distance minimal in this instance, a new Braitenberg coefficient of 1 has been selected, as shown in **figure 3.12(d)**.

3.8 Orbital Mechanics

It is essential in the control of any engineering system to analyse the environment with which the system interacts. The model would not be complete unless a simulation of the effects of the environment on the system was included.

The robot is to be stationed onboard a space station or spacecraft, in orbit around the Earth. The interaction between the spacecraft and Earth may be modelled using the familiar two-body problem, providing an orbiting co-ordinate reference frame with which to analyse the trajectory of the robot. This is essential if an accurate portrayal to the robot's motion whilst in orbit is to be recorded relative to the datum spacecraft within which it exists.

3.8.1 The Two-Body Problem

Many simple orbit simulations are based on the two-body problem, which utilises Newton's second law to determine the forces attracting two bodies and allows a closed form solution to the equations of motion to be found. In order for the two-body problem to be feasible however, certain assumptions must be made as follows:

- The two bodies should be modelled as point masses.
- The only external force acting on the system should be gravity.

Figure 3.14 displays two point masses representing the Earth (m_1) positioned at, $\underline{r}_1$ and the space station (m_2) positioned at $\underline{r}_2$. m_1 and m_2 are separated by vector $\underline{r}$.

Expressing gravity as the inverse square force field and applying Newton's Second Law of Motion for mass m_1:

$$m_1 \ddot{\underline{r}}_1 = G\frac{m_1 m_2}{r^3}\underline{r} \tag{3.25a}$$

and for mass m_2:

$$m_2 \ddot{\underline{r}}_2 = -G\frac{m_1 m_2}{r^3}\underline{r} \tag{3.25b}$$

Given the relationship $\underline{r} = \underline{r}_2 - \underline{r}_1$, the relative acceleration between the two masses can be derived from equations **3.25(a)** and **3.25(b)** as:

$$\ddot{\underline{r}} = -G\frac{m_1 + m_2}{r^3}\underline{r} \qquad (3.25c)$$

If m_1 is much larger than m_2, as is the case here, then it may be said, by expressing the gravitational constant:

$$\mu = Gm_1 \qquad (3.26)$$

(where m_1 is the mass of the Earth (5.976×10^{24}kg) and G is the universal constant ($6.63\times10^{-11}\mathrm{m^3kg^{-2}s}$)) that:

$$\ddot{\underline{r}} = -\mu\frac{\underline{r}}{r^3} \qquad (3.27)$$

which may be written as:

$$\ddot{\underline{r}} + \mu\frac{\underline{r}}{r^3} = 0 \qquad (3.26)$$

Now, in order to incorporate the motion of the robot relative to the orbiting spacecraft, we must firstly discuss relative motion.

3.8.2 Relative Motion

We now introduce a third mass, manoeuvring relative to the space station, m_3. This mass represents the free-flying robot, which is displaced outwith the space station's orbit, and which, in addition to an acceleration due to gravity, has a control acceleration $\underline{a}$. Since these two masses are in motion relative to the Earth in separate orbits, the two-body problem based on an inertially fixed co-ordinate frame is no

longer adequate here. In this case, in order to describe the relative motion, a rotating co-ordinate frame centred on a fixed point on the space station must be introduced.

Figure 3.15 displays the three masses complete with the new co-ordinate frame, from which the equations of relative motion can be developed. Mass m_1 represents the Earth, m_2 represents the space station orbiting the Earth in a circular

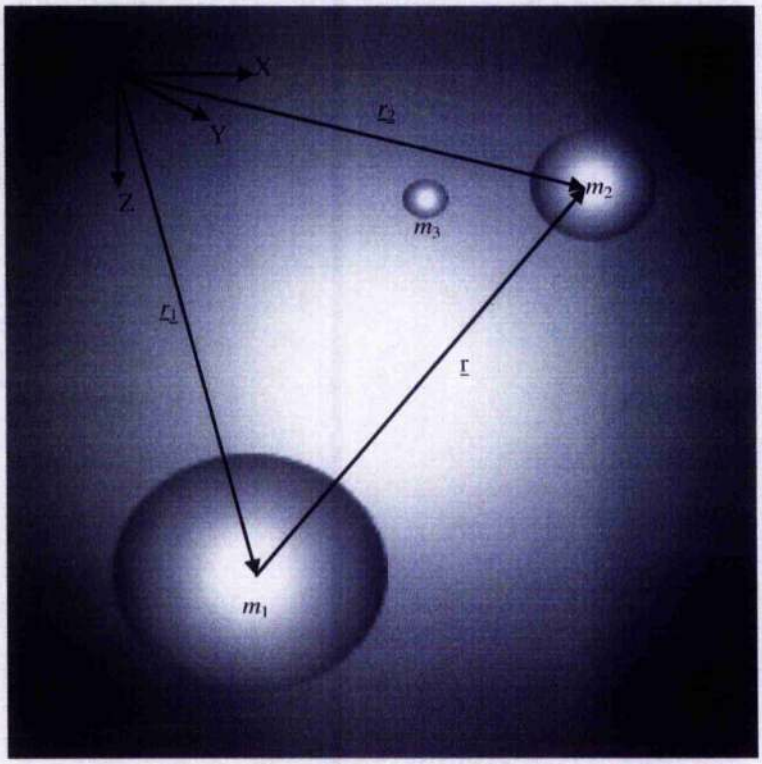


Figure 3.14: The two-body problem displaying two masses m_1 and m_2, separated by vector $\underline{r}$.

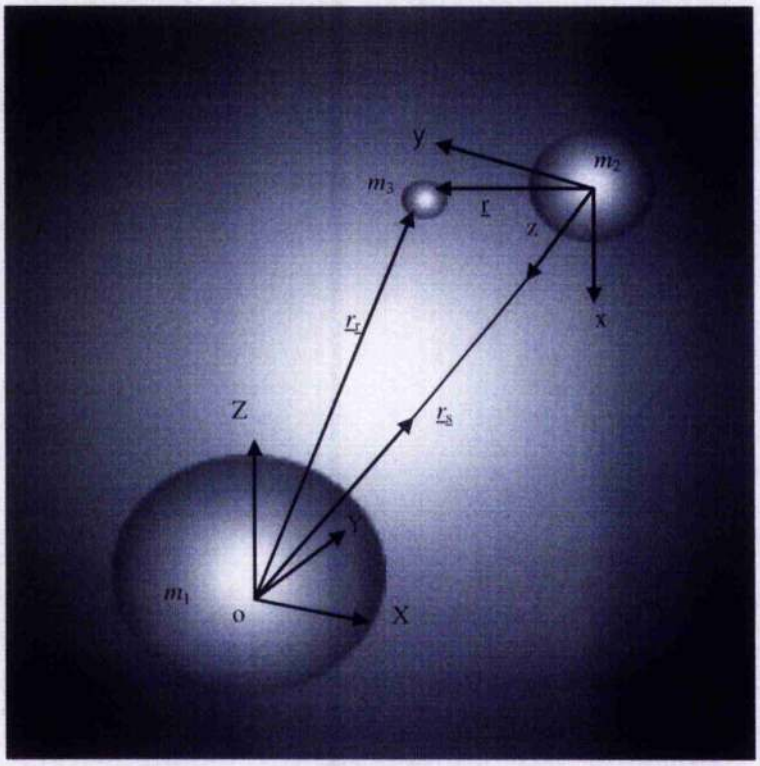


Figure 3.15 Rotating frame of reference fixed on space station denoted m_2.

orbit of radius $\underline{r}_s$ and having its own rotating co-ordinate frame. The mass m_3 represents the robot, displaced from the origin of the rotating co-ordinate frame by distance $\underline{r}$, and displaced from the Earth by a distance $\underline{r}_r$. The forces active on these bodies arise from gravity, however the robot is also influenced by an acceleration $\underline{a}$ resulting from the robot's internal autonomous control system.

From **figure 3.15**, it may be deduced that:

$$\underline{r}_r = \underline{r}_s + \underline{r} \tag{3.27}$$

where $\underline{r}$, the relative position of the robot with respect to the space station, can be given in Cartesian co-ordinates in the rotating frame of reference as:

$$\underline{r} = x\underline{i} + y\underline{j} + z\underline{k} \tag{3.28}$$

Then the position vector of the space station relative to the Earth becomes:

$$\underline{r}_s = -r_s \underline{k} \tag{3.29}$$

and by substituting **equations (3.28)** and **(3.29)** into **equation (3.27)**, the position vector of the robot may be given as:

$$\underline{r}_r = x\underline{i} + y\underline{j} + (z - r_s)\underline{k} \tag{3.30}$$

Since the assumption is made that the only external force acting on the space station is that due to gravity $\underline{g}_s$, then:

$$\ddot{\underline{r}}_s = \underline{g}_s \tag{3.31}$$

and since the robot has an additional control acceleration $\underline{a}$:

$$\ddot{\underline{r}}_r = \underline{g}_r + \underline{a} \tag{3.32}$$

where $\underline{g}_r$ is the acceleration due to gravity acting on the robot. It can be shown that $\underline{g}_r$ may be expressed using direction cosines as:

$$\underline{g}_r = \left(-|g_r|\frac{x}{r_r}\right)\underline{i} + \left(-|g_r|\frac{y}{r_r}\right)\underline{j} + \left(|g_r|\frac{z - r_s}{r_r}\right)\underline{k} \tag{3.33}$$

The orbital angular velocity of the rotating frame, $\underline{\omega}$, is expressed as:

$$\underline{\omega} = \omega \underline{j} \tag{3.34}$$

Now, $\underline{r}$ has been defined as the relative position of the robot with respect to the space station, differentiating with respect to the Earth based co-ordinate system gives:

$$\frac{d}{dt}\underline{r}=\frac{\partial}{\partial t}\underline{r}+\underline{\omega}\times\underline{r} \tag{3.35}$$

Where ∂ symbolises differentiation with respect to the rotating frame, and $\underline{\omega}$ represents the angular velocity of the rotating frame.

Differentiating further gives the relative angular acceleration as:

$$\frac{d^2}{dt^2}\underline{r}=\frac{\partial^2}{\partial t^2}\underline{r}+2\left(\underline{\omega}\times\dot{\underline{r}}\right)+\dot{\underline{\omega}}\times\underline{r}+\underline{\omega}\times(\underline{\omega}\times\underline{r}) \tag{3.36}$$

Combining **equation (3.36)** with above **equation (3.27)** allows the acceleration of the robot to be calculated as:

$$\ddot{\underline{r}}_r=\ddot{\underline{r}}_t+\ddot{\underline{r}}+2\left(\underline{\omega}\times\dot{\underline{r}}\right)+\dot{\underline{\omega}}\times\underline{r}+\underline{\omega}\times(\underline{\omega}\times\underline{r}) \tag{3.37}$$

and finally using **equation (3.37)** with **(3.34)**, results in the following non-linear differential equations of motion:

$$\ddot{x}=-g_r r\left(\frac{x}{r_r}\right)+a_x+2\omega\dot{z}+\dot{\omega}z+\omega^2 x \tag{3.39}$$

$$\ddot{y}=-g_s\left(\frac{y}{r_r}\right)+a_y \tag{3.40}$$

$$\ddot{z}=+g_r\left(\frac{z\omega r_s}{r_r}\right)+a_z-g_s-2\omega\dot{x}-\dot{\omega}x+\omega^2 z \tag{3.41}$$

Now, it is assumed that the distance between the space station and the Earth is much larger than the distance between the space station and the robot such that:

$$|r_s|>>|r| \tag{3.42}$$

Then **equations (3.39)** through **(3.41)** may be linearised. These linearised equations are known as the *Clohessy-Wiltshire* equations (Clohessy & Wiltshire, 1960) and may be obtained through the following procedure:

Firstly, approximating:

$$r=\left[x^2+y^2+\left(z+r_s\right)^2\right]^{1/2}\approx r_s\left(1+\frac{z}{r_s}\right) \tag{3.43}$$

$$g=\frac{g_s r_s^2}{r^2}\approx g_s\left(1-\frac{2z}{r_s}\right) \tag{3.44}$$

$$-g\frac{x}{r}\approx -g_s\frac{x}{r_s} \tag{3.45}$$

$$-g\frac{y}{r}\approx -g_s\frac{y}{r_s} \tag{3.46}$$

$$-g\left(\frac{z+r_s}{r}\right)\approx -g_s\left(1-\frac{2z}{r_s}\right) \tag{3.47}$$

and so, **equations (3.39)** through **(3.41)** may be linearised giving:

$$\ddot{x}=-g\frac{x}{r}+a_x+2\omega\dot{z}+\dot{\omega}z+\omega^2x \tag{3.48}$$

$$\ddot{y}=-g\frac{y}{r}+a_y \tag{3.49}$$

$$\ddot{z}=+g\left(1+\frac{2z}{r}\right)+a_z-g-2\omega\dot{x}-\dot{\omega}x+\omega^2z \tag{3.50}$$

If the space station is in a circular orbit however:

$$\omega = \sqrt{\frac{g}{r}} \tag{3.51}$$

where:

$$\dot{\omega} = 0 \tag{3.52}$$

and considering the unforced case where $a_x = a_y = a_z = 0$, **equations (3.48) – (3.50)** reduce further to:

$$\ddot{x} - 2\omega\dot{z} = 0 \tag{3.53}$$

$$\ddot{y} + \omega^2 y = 0 \tag{3.54}$$

$$\ddot{z} - 3\omega^2 z + 2\omega\dot{x} = 0 \tag{3.55}$$

This set of equations is commonly known as the Clohessy-Wiltshire equations, which are used extensively in space applications. On integration they yield the motion of the robot relative to the space station.

Equation (3.54) may be expressed as a simple harmonic oscillator given as:

$$y(t) = y_o \cos(\omega t) + \frac{\dot{y}_o}{\omega}\sin(\omega t) \tag{3.56}$$

and differentiating this expression gives:

$$\dot{y}(t) = -y_o \omega \sin(\omega t) + \dot{y}_o \cos(\omega t) \tag{3.58}$$

From this, given the robot's initial velocity and position, the y position and velocity terms relative to the origin of the space station co-ordinate system can be calculated. The x and z components are however coupled. Their velocity and position may be calculated though by introducing a forcing term to a simple harmonic oscillator:

$$x(t) = -2\frac{\dot{z}_o}{\omega}\cos(\omega t) + \left(4\frac{\dot{x}_o}{\omega} - 6z_o\right)\sin(\omega t) - \left(3\dot{x}_o - 6\omega z_o\right)t + x_o + \frac{2\dot{z}_o}{\omega} \qquad (3.59)$$

and by differentiating, this gives:

$$\dot{x}(t) = -2\dot{z}_o\sin(\omega t) + \left(4\dot{x}_o - 6\omega z_o\right)\cos(\omega t) - \left(3\dot{x}_o - 6\omega z_o\right) \qquad (3.60)$$

The z-component velocity and position are likewise found to be:

$$z(t) = 4z_o - 2\left(\frac{\dot{x}_o}{\omega}\right) + \left(\frac{\dot{z}_o}{\omega}\right)\sin(\omega t) + \left(2\frac{\dot{x}_o}{\omega} - 3z_o\right)\cos(\omega t) \qquad (3.61)$$

$$\dot{z}(t) = \dot{z}_o\cos(\omega t) + \left(3\omega z_o - 2\dot{x}_o\right)\sin(\omega t) \qquad (3.62)$$

Figure 3.16 traces the slow drift motion of the robot relative to the space station when the robot is offset from the origin of the space station co-ordinate frame in all three axes. The robot is initially located 4 metres from the centre of the space station on each axis, and is given an initial relative velocity of zero ms^{-1}. Over a period of 1000 seconds, the robot drifts from its initial start position to find itself displaced some distance away at the end of the simulation. This is due to the centre of mass of the robot and the centre of mass of the space station being in different orbits around the Earth, and so, there will over time, be a displacement relative to each other, with respect to the Earth.

Figure 3.17 shows this motion plotted separately on each axis whilst superimposing thruster activity to demonstrate that this free drift occurs with zero control acceleration and is indeed due to the influence of being in slightly different orbits. The robot in this example is offset into a larger orbit than the centre of the

space-station, therefore, it would appear to an observer on an inertial axis to be moving slower than the space-station. This can be seen from **figures (3.16) – (3.18).**

3.9 Discussion

This chapter followed the design, development and testing of a simple inertial control system, based on Braitenberg Theory, to provide an initial basis for easy analysis, and systematic prediction of the robot's behaviour in numerous situations before expanding into a more integrated control system.

A collision avoidance algorithm formed the basis of this controller by using the robots' virtual proximity sensors to directly measure the distance to obstacles lying along the sensors' line-of-sight and within their horizon, thus enabling collision avoidance by providing a simple estimation of, and reaction to, the environment. The collision avoidance acceleration was then filtered to pulse thrusters when a threshold value was reached to realistically mimic actual thruster activity. The model was then expanded by introducing a wall following algorithm later in the chapter to aid the robot in finding its way more easily in a constrained environment by following wall surfaces encountered, and then enhanced by the addition of an appropriate sensor horizon for the chosen sensor type. Incorporating the dynamics of the free-flyer relative to the orbiting spacecraft demonstrating the slow drift influence from parallel orbits and completed the model. The model was tested at each level of design and from test results, appropriate variables chosen to ensure an efficient working model.

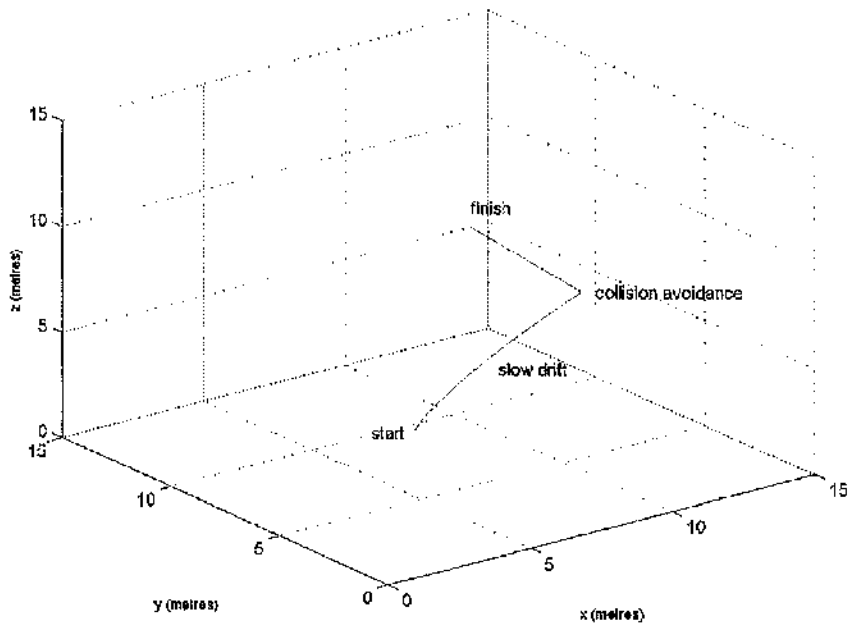


Figure 3.16: With the robot offset from the spacecraft in all three axes, the plot depicts the slow drift of the robot relative to the space station co-ordinate frame as they both co-orbit the Earth.

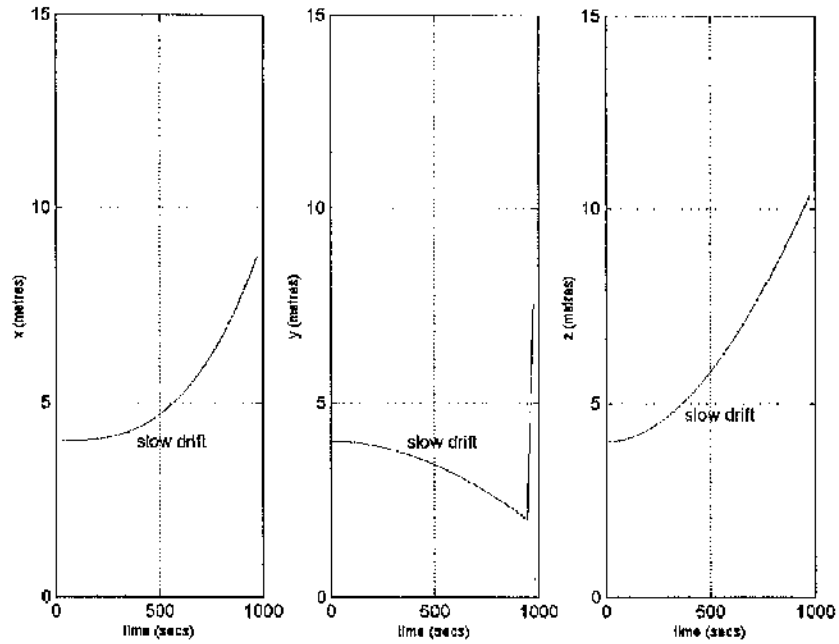


Figure 3.17: Slow drift motion of the robot with respect to the space-station co-ordinate frame plotted on each axis over a period of 1000 seconds.

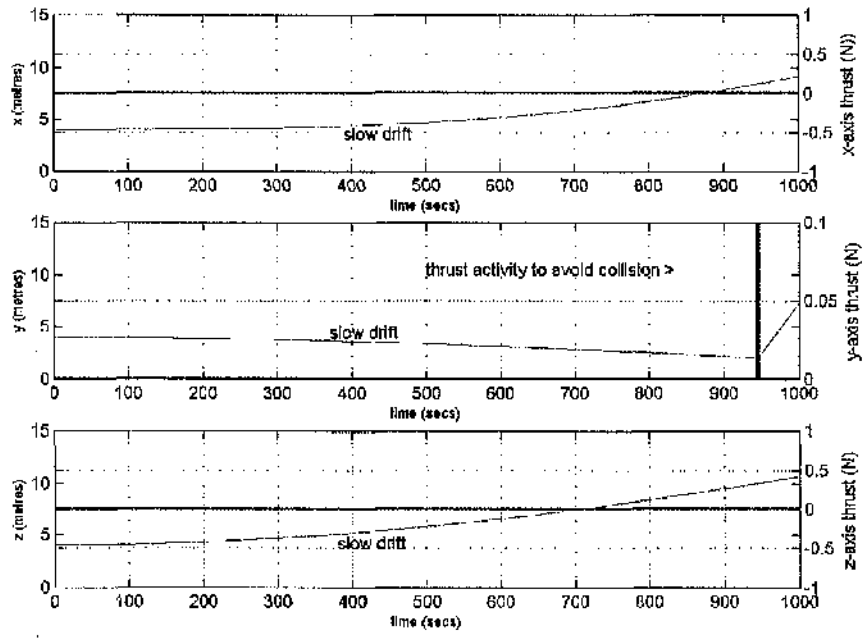


Figure 3.18: Slow drift motion of the robot with respect to the space-station coordinate frame plotted on each axis over a period of 1000 seconds, super-imposing thrust activity.

Chapter Four: The Velocity Control System

'Who are you?
Who slips into my robot shell
and whispers to my ghost?'

Ghost in the Shell

4.1 Introduction

The initial control methodology developed in **chapter three** has been shown to successfully perform the tasks required. However, it is simplistic in design and limited in capabilities. Improving and expanding this simple controller, to develop a more realistic and versatile model, therefore, becomes the next logical step.

In **chapter three**, the thrusters act along inertially fixed axes and collisions are avoided by employing thruster activity to directly control the free-flyer's acceleration. This results in a motion that can be described as elastic, displaying sharp changes in direction resulting from acceleration changes. The robot in this situation only entertains translational motion in straight lines. Thrust acts only through the robot's centre of mass, and so no moment is produced. The simplicity of this design is expanded here to improve motion versatility, and allow for rotational activity to become incorporated in later chapters.

This chapter takes the analysis to the next phase by converting the acceleration control system into a *velocity control system*. The system maintains a constant total velocity magnitude, however collisions are avoided by 'bending' the velocity vector by changing the velocity components, manifest through appropriate thruster firing. The origin of the axes of the velocity components lies in the centre of mass of the robot and the robot will ultimately rotate with it.

4.2 Velocity Control

To create this velocity control system, our initial design must force the total velocity magnitude, so far only influenced by collision avoidance strategies $|\underline{v}_{av}|$, to remain constant, which can be done by normalisation techniques. By only changing velocity *components* on each plane enables collisions to be avoided whilst following a smooth, curved trajectory.

Again, as with **chapter 3**, the control system is driven by the sensor outputs from each plane of detectors, and again, an inverse relationship between the information from each sensor and the distance to the obstacle is formed to give a resultant signal strength S_i for each sensor i.

$$S_i = \frac{1}{d_i} \qquad i=1\text{-}6 \tag{4.1}$$

where d_i is the distance recorded by sensor i.

However, in contrast to **chapter three**, the two sensor signals on each plane are combined to directly calculate the resultant *velocity* required to avoid colliding with the obstacle. Again, coefficients are introduced to vary the weighting of each sensor. This is calculated as:

$$\begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = \begin{bmatrix} c_1 & c_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & c_3 & c_4 & 0 & 0 \\ 0 & 0 & 0 & 0 & c_5 & c_6 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \end{bmatrix} \tag{4.2}$$

Where v_x, v_y and v_z are the robot velocity components desired on each axis and c_i are the Braitenberg coefficients. This may be summarised as:

$$\underline{v}_{av} = \underline{M} \cdot \underline{S} \tag{4.3}$$

where matrix **M** is a matrix representation of the behaviour and can be controlled by altering the values of the Braitenberg coefficients and $\underline{v}_{av}$ is the robot velocity commanded:

$$v_x = c_1 s_1 + c_2 s_2 \tag{4.4a}$$

$$v_y = c_3 s_3 + c_4 s_4 \tag{4.4b}$$

$$v_z = c_5 s_5 + c_6 s_6 \tag{4.4c}$$

where s_{1-6} are the sensor outputs obtained from $\frac{1}{d_{1-6}}$.

The signal increases as the robot approaches a collision hazard and the desired velocity in each plane is then calculated proportional to the corresponding sensor.

4.3 Calculation of the Command Acceleration

Now, to determine the thruster firing required to avoid obstacles, a command acceleration is calculated as follows:

A unit normal pointing along the velocity vector may be found from:

$$\underline{n} = \frac{\underline{v}_{av}}{|\underline{v}_{av}|} \tag{4.5}$$

Expressing the unit normal in component form:

$$\underline{n} = [n_x \quad n_y \quad n_z] \tag{4.6}$$

which may then be expressed as:

$$\underline{n} = \left[\frac{v_x}{v_{av}} \quad \frac{v_y}{v_{av}} \quad \frac{v_z}{v_{av}} \right] \tag{4.7}$$

The components of the above unit normal may be expressed from **equations 4.4(a)-(c)** as:

$$\underline{n}_x = \frac{v_x}{\sqrt{v_x^{\,2} + v_y^{\,2} + v_z^{\,2}}} \tag{4.8a}$$

$$\underline{n}_y = \frac{v_y}{\sqrt{v_x^{\,2} + v_y^{\,2} + v_z^{\,2}}} \tag{4.8b}$$

$$\underline{n}_z = \frac{v_x}{\sqrt{v_z^{\,2} + v_y^{\,2} + v_z^{\,2}}} \tag{4.8c}$$

Now, the required acceleration is obtained from:

$$\underline{a}_{av} = \frac{d}{dt} \underline{v}_{av} \tag{4.9}$$

which, in each direction, is derived from **equation 4.7** as:

$$\underline{a}_{av} = \frac{d}{dt} \underline{v}_{av} = \frac{d}{dt} \left(|\underline{v}_{av}| \underline{n} \right) \tag{4.10}$$

which then becomes:

$$a_{avx} = v_{av} \dot{n}_x + \dot{v}_{av} n_x \tag{4.11a}$$

$$a_{avy} = v_{av}\dot{n}_x + \dot{v}_{av}n_y \tag{4.11b}$$

$$a_{avz} = v_{av}\dot{n}_z + \dot{v}_{av}n_z \tag{4.11c}$$

Since v_{av} is a constant, $\dot{v}_{av} = 0$, therefore:

$$a_{avx} = v_{av}\dot{n}_x \tag{4.12a}$$

$$a_{avy} = v_{av}\dot{n}_y \tag{4.12b}$$

$$a_{avz} = v_{av}\dot{n}_z \tag{4.12c}$$

By substituting **equations 4.8(a)-(c)** into **equations 4.12(a)-(c)**, $a_{avx}, a_{avy}, a_{avz}$ then become:

$$a_{avx} = v_{av}\frac{d}{dt}\left(\frac{v_x}{\sqrt{v_x^2 + v_y^2 + v_z^2}}\right) \tag{4.13a}$$

$$a_{avy} = v_{av}\frac{d}{dt}\left(\frac{v_y}{\sqrt{v_x^2 + v_y^2 + v_z^2}}\right) \tag{4.13b}$$

$$a_{avz} = v_{av}\frac{d}{dt}\left(\frac{v_x}{\sqrt{v_z^2 + v_y^2 + v_z^2}}\right) \tag{4.13c}$$

This may be written as:

$$a_{avx} = v_{av}\left(\frac{\dot{v}_x}{\sqrt{v_x^2 + v_y^2 + v_z^2}} - \frac{v_x\left(v_x\dot{v}_x + v_y\dot{v}_y + v_z\dot{v}_z\right)}{\left(\sqrt{v_x + v_y + v_z}\right)^3}\right) \tag{4.14a}$$

$$a_{avy} = v_{av}\left(\frac{\dot{v}_y}{\sqrt{v_x^2+v_y^2+v_z^2}} - \frac{v_y\left(v_x\dot{v}_x+v_y\dot{v}_y+v_z\dot{v}_z\right)}{\left(\sqrt{v_x+v_y+v_z}\right)^3}\right) \tag{4.14b}$$

$$a_{avz} = v_{av}\left(\frac{\dot{v}_z}{\sqrt{v_x^2+v_y^2+v_z^2}} - \frac{v_z\left(v_x\dot{v}_x+v_y\dot{v}_y+v_z\dot{v}_z\right)}{\left(\sqrt{v_x+v_y+v_z}\right)^3}\right) \tag{4.14c}$$

Where from **equation (4.4)**:

$$\dot{v}_x = -\frac{c_1\dot{d}_1}{d_1^2} - \frac{c_2\dot{d}_2}{d_2^2} \tag{4.15a}$$

$$\dot{v}_y = -\frac{c_3\dot{d}_3}{d_3^2} - \frac{c_4\dot{d}_4}{d_4^2} \tag{4.15b}$$

$$\dot{v}_z = -\frac{c_5\dot{d}_5}{d_5^2} - \frac{c_6\dot{d}_6}{d_6^2} \tag{4.15c}$$

This series of equations enables collision avoidance by manipulating the robot's velocity vector to bend its trajectory away from obstacles whilst maintaining the total velocity magnitude as a constant; this is enabled by only changing the *components* of the velocity vector, whilst the total velocity is forced to remain the same, thus causing a bending of the velocity vector as the components change in magnitude. **Equations 4.15(a)** to **(c)** result in the fixing of the total velocity magnitude by altering only the velocity components by normalisation techniques as derived through equations 4.7 to 4.14, which keep the total velocity constant, to enable a smooth, curved trajectory. These equations replace **equation 3.3** within the simulation program, and as with **chapter three**, the change in each velocity component is calculated from the new accelerations as:

$$\Delta v_i = a_{av_i}\Delta t \tag{4.16}$$

where i is defined as x, y, z as in **equation (3.1)**.

Taking this expression, the new velocity components are then calculated using the Euler integration method:

$$v_i = v_{i_{prev}} + \Delta v_i \tag{4.17}$$

where again i is defined as x, y, z and the robot position is updated using a similar scheme to **equation 3.6**. By employing velocity magnitude fixing allows the robot to bend away from collision hazards. In the instance of the robot headed straight for a wall, the total magnitude of the velocity remains the same, however, the components change to alter the direction of the robot, enabling collision avoidance in a more versatile manner, as shown in figure 4.1 below.

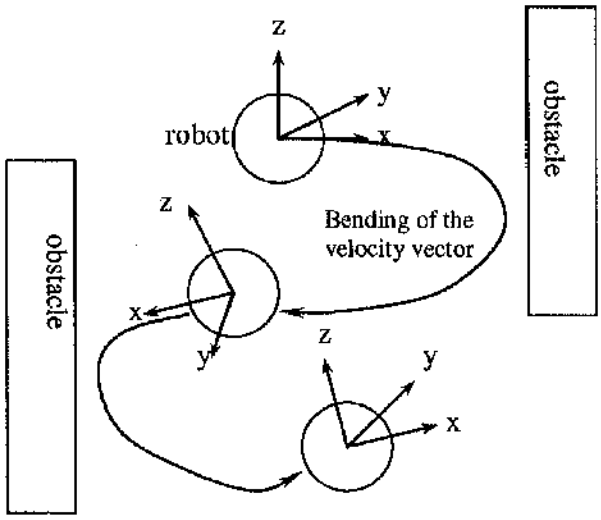


Figure 4.1: Bending of the velocity vector keeps the total velocity magnitude constant.

4.4 Testing the Velocity Controller

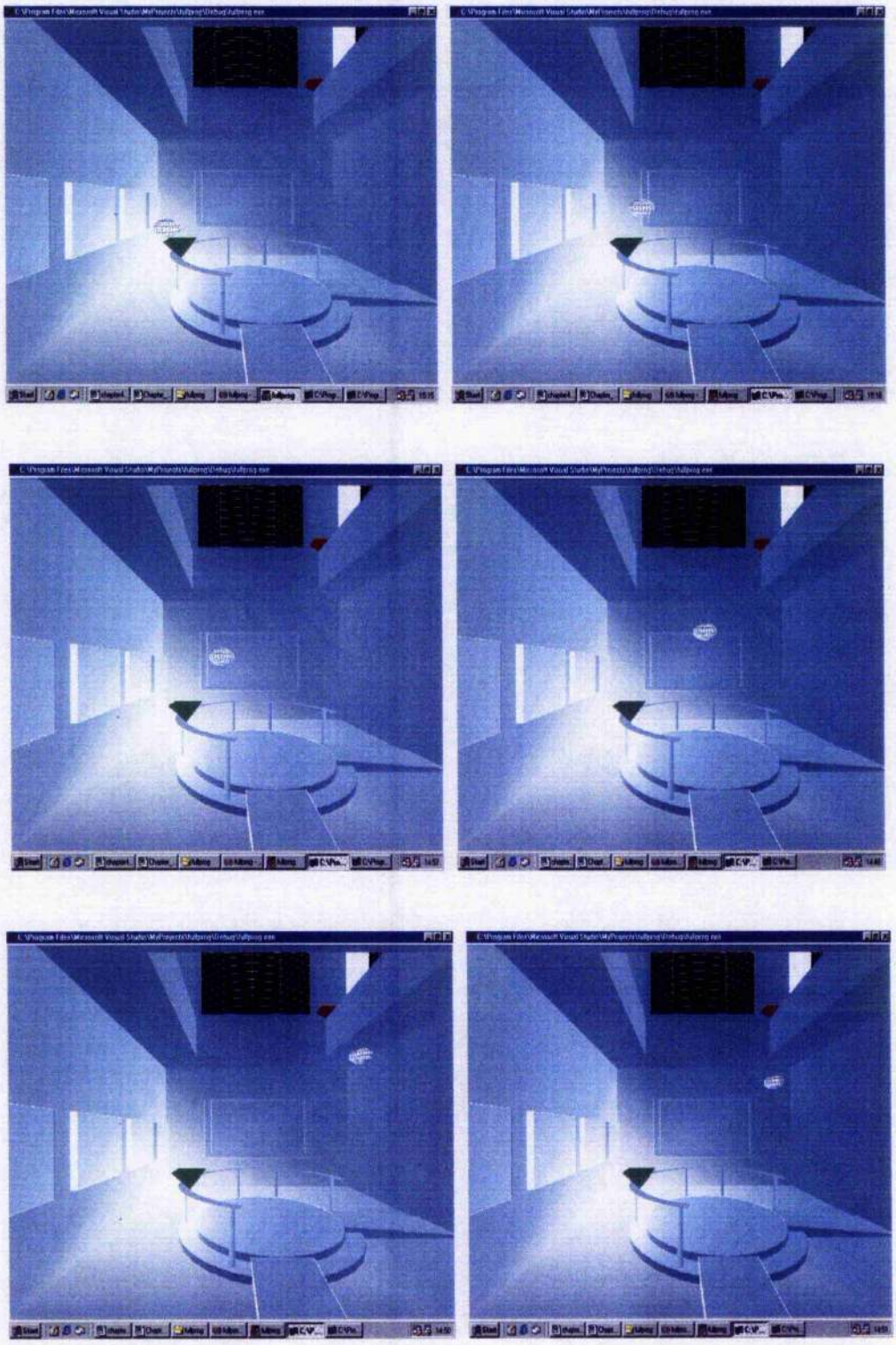
Figure 4.2 shows a series of still frames capturing the motion of the free-flyer manoeuvring in its virtual environment during a run of the computer simulation and enforcing collision avoidance firstly at the top right -> then top left -> then bottom left of the workspace.

Note also from these still frames that the environment has been enhanced from **chapter three** to provide more collision hazards to aid in testing the free-flyer algorithms.

As in **chapter three**, the matrix of Braitenberg constants for this initial run is again:

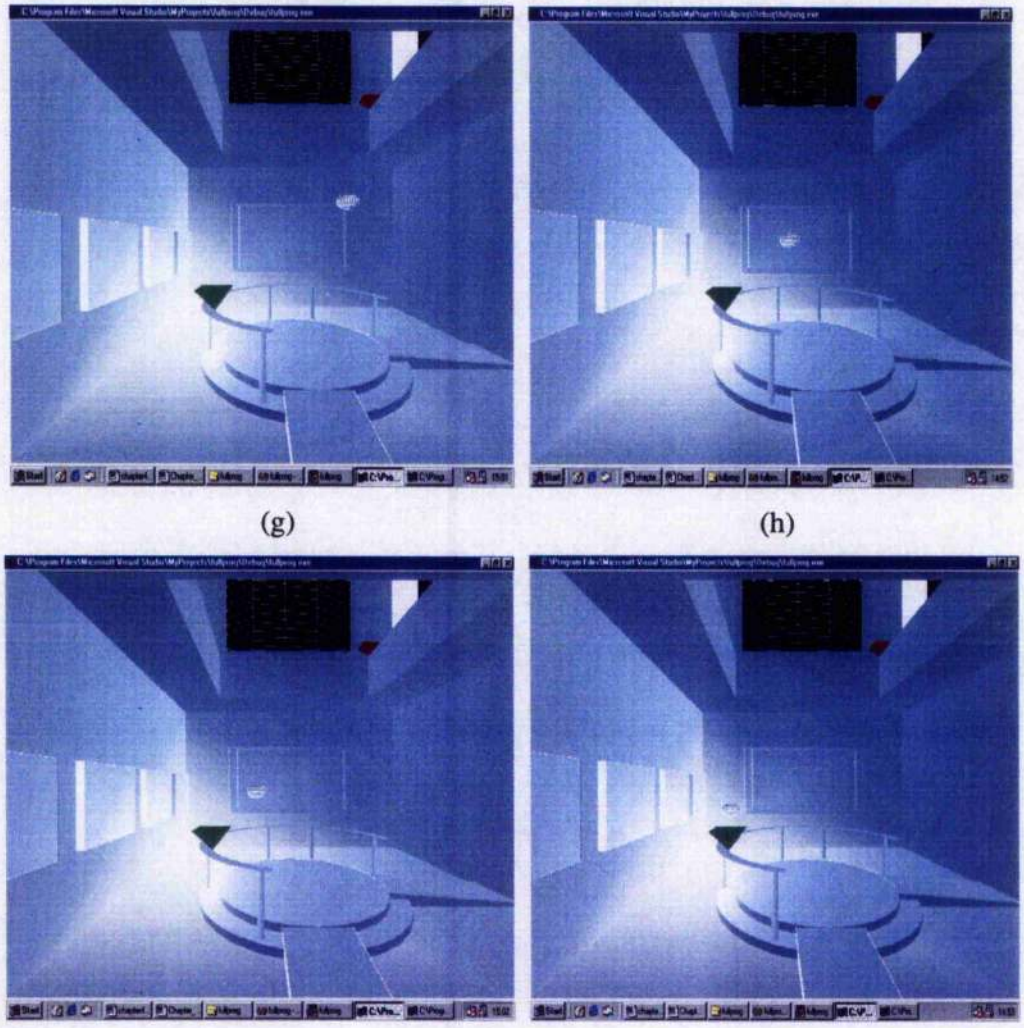
$$M = \begin{bmatrix} 0.5 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & 0.5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0.5 \end{bmatrix} \tag{4.18}$$

The snapshots, as before, capture the obstacle avoidance algorithm over a period of 100 seconds and aid in the perception of the free-flyer's motion and interaction with its three-dimensional environment.



(e) (f)

Figures 4.2(a) - (j): Frames capture the motion of the robot.



(g) (h)

(i) (j)

Figures 4.2(a) - (j): Frames capture the motion of robot.

Testing the algorithm begins in **figure 4.3**, which traces the trajectory of the robot when employing the method of velocity control over a period of 300 seconds. **Figure 4.4** plots the position components on each axis against time over this same time period.

Figure 4.3 shows clearly the bending of the velocity vector on approach to obstacles, consequential from the use of the velocity controller, which may be contrasted with **figure 3.4,** which demonstrates the sharp changes in direction achieved from using the initial acceleration controller designed and employed in **chapter three**. Velocity control, as demonstrated in **figure 4.3** is shown to become

the next obvious step by increasing the versatility of the model. The motion progresses from purely translational motion in straight lines as in **chapter 3**, to curved motion here in **chapter 4**, and from this to incorporate rotational motion, as followed in **chapter 5**.

4.5 Discussion

This chapter has seen the progression of the model from a simple system whereby the trajectory is purely translational and the robot recoils from obstacles in a sharp elastic manner, to one that incorporates bending of the velocity vector, providing a smooth curved rotation from obstacles. This required fixing the total velocity magnitude and altering only velocity components by normalisation techniques - to enable a smooth, curved trajectory.

By enhancing the system through increasing trajectory versatility paves the way for the incorporation of rotational motion and allowing for camera fixed pointing, which will be seen in the following chapter.

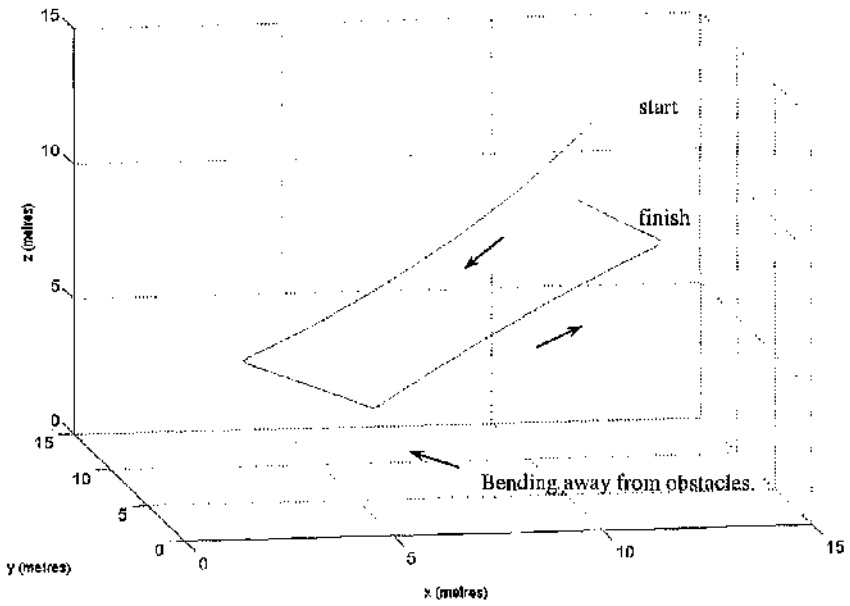


Figure 4.3: The bending of the robot trajectory away from obstacles from incorporating the velocity control method.

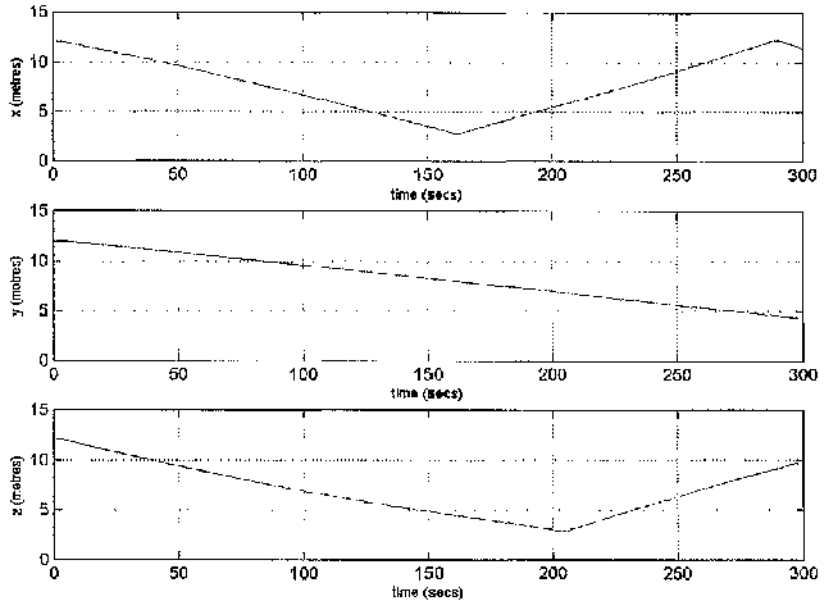


Figure 4.4: x, y & z components of vehicle trajectory shown in **figure 4.3.**

Chapter Five: Camera Fixed Pointing

'Do androids dream of electric sheep?'

Philip K. Dick

5.1 Introduction

Chapters 3 and **4** have seen the development of a robot translational system, capable of manoeuvring autonomously in an unknown environment whilst successfully avoiding obstacles.

In this chapter, development is continued to provide the robot with some function. If the robot is now assumed to be fitted with a camera, therein lies the possibilities for video conferencing, by requesting the camera optical axis to be fixed on some desired point, with the robot orientating appropriately to camera track the object (or astronaut) of interest, with either, or both, the camera and the object in motion.

The chapter begins by developing a camera simulation tool to depict the view that would be seen though the camera lens, and then continues with the development of an attitude control system which would bring the camera optical axis to rest in the desired direction, or track a moving line of sight.

Providing the robot with a camera and incorporating camera fixed pointing into the control system, provides the robot with a function, which would clearly be desirable onboard a crewed vehicle.

5.2 The Camera Simulation Tool

A second viewing window was integrated into the simulation using the OpenGL® software interface. This window provides a virtual view though the camera lens and acts in first person, moving with the robot, in addition to the existing third person viewing window discussed in **chapter 2.**

The viewing volume chosen for the virtual camera lens has a field of view of 90 degrees, giving a large scope of view. The *near* and *far* values (distances between the viewpoint and the clipping planes) are given the values 1 metre and 16 metres respectively, to allow the full depth of the module to be viewed from any point (since the length, width and height of each module is 15 metres. An aspect ratio (ratio of width to height of the viewing plane) is chosen as 1, to avoid distorting the image.

Figure 5.1 demonstrates the perspective viewing volume that was specified using the OpenGL Utility Library (GLU) to produce the interactive three-dimensional application, whilst **figures 5.2 (a) - (d) display** screenshots taken through the camera tool, of the surrounding environment.

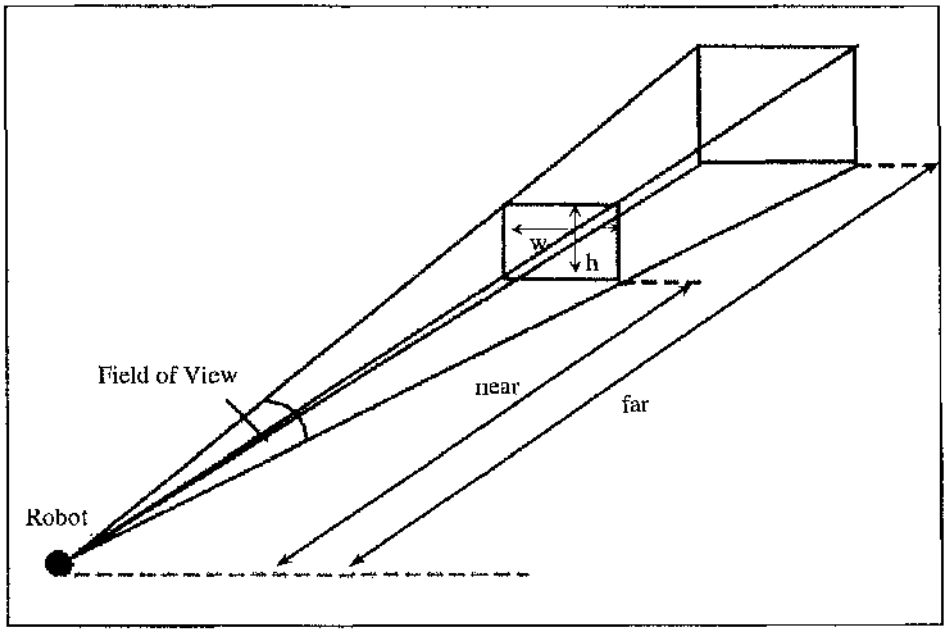
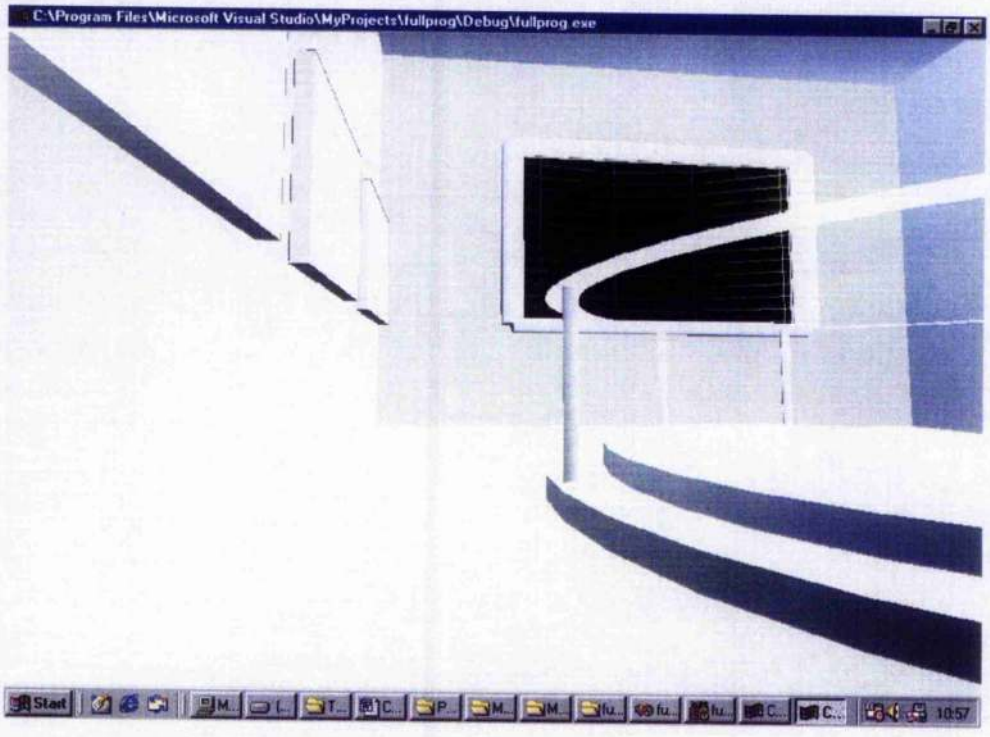


Figure 5.1: Viewing volume perspective, with aspect rule w/h..

(a)



(b)

Figure 5.2 Snapshots of the virtual environment as seen through the camera tool.

(c)

(d)

Figure 5.2 Snapshots of the virtual environment as seen through the camera tool.

The camera simulation tool is first given an initial camera direction, chosen here to be (0, 1, 0) i.e. pointing along the robot body's y-axis. This was chosen since the co-ordinate system employed by OpenGL® for the environment has the y-axis pointing ahead, and the most logical starting base for the camera orientation would be to point it straight ahead, as illustrated in **figure 5.3**. The robot is represented as a sphere, with camera body axes defined as X_r, Y_r and Z_r.

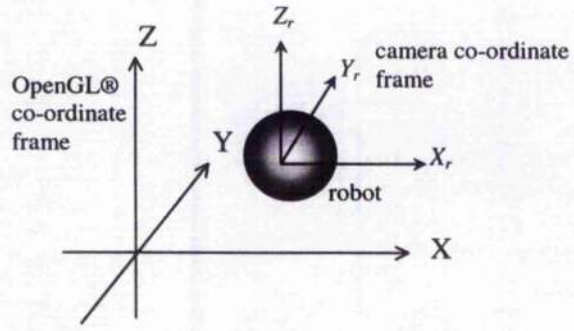


Figure 5.3: Reference frames of the camera and space station.

5.3 Developing the Camera Simulation Tool

Now, to develop the camera tool to enable fixed pointing, the tool is given an initial viewing target position based on the mission requirements. It is required that at each time-step, the control system would compare its actual camera direction with the required target direction and then provide appropriate measures to achieve the desired camera pointing direction. These measures are explained and developed in the following sections:

5.3.1 Orientation Angles and Transformations of Co-ordinates

To describe the orientation of one body with respect to another, a reference frame and a system of three angles is required. There are a number of possible choices, the most popular of which is the Euler transformation using the Euler angles.

5.3.2 *The Euler Rotations*

A commonly used method for orienting a rigid body to a desired attitude is called *body-axis rotation*; it involves rotating the body-fixed reference frame three times about a set of axes. The first rotation may take place around any axis, the second should take place around either of the two axes not used in the first rotation, and the third rotation takes place around either of the two axes not used in the second rotation (Murphy, 1993), (Meriam & Kraig, 1993), (James, 1994). This concept of using three successive rotations to describe the orientation of an orbit plane was first introduced by Leonard Euler (1707-1783) (Euler, 1988), (Dunham, 1999), (Simmons, 1996).

Consider three successive body-axis rotations describing the orientation of a reference frame ($\underline{i}$, $\underline{j}$, $\underline{k}$) relative to a body frame ($\underline{b}_1$, $\underline{b}_2$, $\underline{b}_3$). In order to align the reference frame with the body frame of the robot, three rotations (yaw, pitch and roll) take place around the z, y and x-axes, respectively.

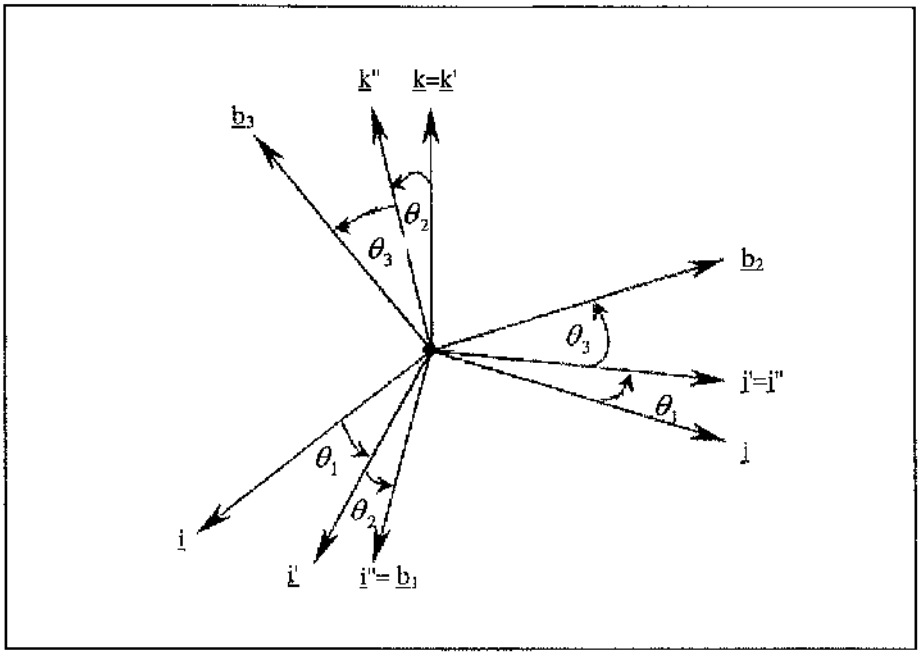


Figure 5.4: Euler angle definition.

First, a rotation of angle θ_1 about vertical axis **i** (roll), followed by a rotation of θ_2 about intermediate axis **j'** (pitch), and finally a rotation of angle θ_3 about **k''** (yaw) as shown in **figure 5.4**. The body frame of reference is denoted $\mathbf{b_1}$, $\mathbf{b_2}$, $\mathbf{b_3}$, and observe also, that **i''** is a unit vector in the body frame, specifically, $\mathbf{b_1}$.

If unit axes **i'**, **j'**, **k'** are rotated through angle θ_1 relative to reference frame **i**, **j**, **k**, then the components of **i**, **j**, **k** along **i'**, **j'**, **k'** are:

$$\mathbf{i'} = \mathbf{i} \tag{5.3a}$$

$$\mathbf{j'} = \cos(\theta_1)\mathbf{j} + \sin(\theta_1)\mathbf{k} \tag{5.3b}$$

$$\mathbf{k'} = -\sin(\theta_1)\mathbf{i} + \cos(\theta_1)\mathbf{k} \tag{5.3c}$$

which can be easily verified by the geometric relations obtained from **figure 5.4**. In matrix notation, this can be expressed as:

$$\begin{bmatrix} i' \\ j' \\ k' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_1 & \sin\theta 1 \\ 0 & -\sin\theta_1 & \cos\theta_1 \end{bmatrix} \begin{bmatrix} i \\ j \\ k \end{bmatrix} = R(\theta_1) \begin{bmatrix} i \\ j \\ k \end{bmatrix} \tag{5.4}$$

where $R(\theta_1)$ is the rotation matrix.

Now, the second and third rotations may be expressed from the same geometric relations as follows:

$$R(0_2) = \begin{bmatrix} \cos(\theta_2) & 0 & -\sin(\theta_2) \\ 0 & 1 & 0 \\ \sin(\theta_2) & 0 & \cos(\theta_2) \end{bmatrix} \tag{5.5}$$

$$R(\theta_3) = \begin{bmatrix} \cos(\theta_3) & \sin(\theta_3) & 0 \\ -\sin(\theta_3) & \cos(\theta_3) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.6}$$

The general transformation of the components of a vector from the reference frame to the body frame may then be obtained from the matrix product:

$$R_{123} = \underline{R}_3(\theta_3)\underline{R}_2(\theta_2)\underline{R}_1(\theta_1) \tag{5.7}$$

so that the composite rotation matrix can be written as:

$$R_{123} = \begin{bmatrix} C\theta_2 C\theta_3 & C\theta_1 S\theta_3 + S\theta_1 S\theta_2 C\theta_3 & S\theta_1 S\theta_3 - C\theta_1 S\theta_2 C\theta_3 \\ -C\theta_2 S\theta_3 & C\theta_1 C\theta_3 - S\theta_1 S\theta_2 S\theta_3 & S\theta_1 C\theta_3 + C\theta_1 S\theta_2 S\theta_3 \\ S\theta_2 & -S\theta_1 C\theta_2 & C\theta_1 C\theta_2 \end{bmatrix} \tag{5.8}$$

where C = cosine; S = sine.

The angular velocity vector of the robot in body axes can now be determined from the geometrical relations shown in **figure 5.4** and the transformation equations. The following components are found:

$$\omega_1 = \dot{\theta}_3 - \dot{\theta}_1 \sin(\theta_2) \tag{5.9}$$

$$\omega_2 = \dot{\theta}_2 \cos(\theta_3) + \dot{\theta}_1 \cos(\theta_2)\sin(\theta_3) \tag{5.10}$$

$$\omega_3 = -\dot{\theta}_2 \sin(\theta_3) + \dot{\theta}_1 \cos(\theta_2)\cos(\theta_3) \tag{5.11}$$

and the inverse relationship provides an expression for the rate of change of the three orientation angles of the robot in terms of its angular velocity components as:

$$\dot{\theta}_1 = \frac{\omega_2 \sin(\theta_3) + \omega_3 \cos(\theta_3)}{\cos(\theta_2)} \tag{5.12}$$

$$\dot{\theta}_2 = \omega_2 \cos(\theta_3) - \omega_3 \sin(\theta_3) \tag{5.13}$$

$$\dot{\theta}_3 = \omega_1 + (\omega_2 \sin(\theta_3) + \omega_3 \cos(\theta_3)) \tan(\theta_2) \tag{5.14}$$

A singularity arises using these orientation angles when pitch angle reaches $\pi/2$. Quaternions could be used to avoid this. Quaternions replace three consecutive rotations about the three orthogonal unit vectors with one single rotation about an eigenvector with unit eigenvalue, which would eliminate the occurrence of singularities arising. Quaternions, however, are non-intuitive to visualise, and so, for ease of illustration, the method using the Euler equations will remain (Wie, 1998), (Wiesel, 1997).

For the purpose of this investigation however, singular orientation can be avoided by a suitable choice of target pointing angles. However, a flight vehicle would require a quaternion-based calculation.

5.4 Orienting the Camera

To point the camera towards a desired target point, a unit vector between the target and robot is required. **Figure 5.5** displays the position of the robot and the target with respect to the origin of inertial axes and shows the unit vector linking them.

Given the target position vector $\underline{r}_t$ with co-ordinates (x_t, y_t, z_t), and the robot, position vector $\underline{r}_r$, with co-ordinates (x_r, y_r, z_r), then the unit vector (N_x, N_y, N_z) which would point the robot towards the target may be calculated as:

$$\underline{N} = \frac{\underline{r}_t - \underline{r}_r}{\left|\underline{r}_t - \underline{r}_r\right|} \tag{5.15}$$

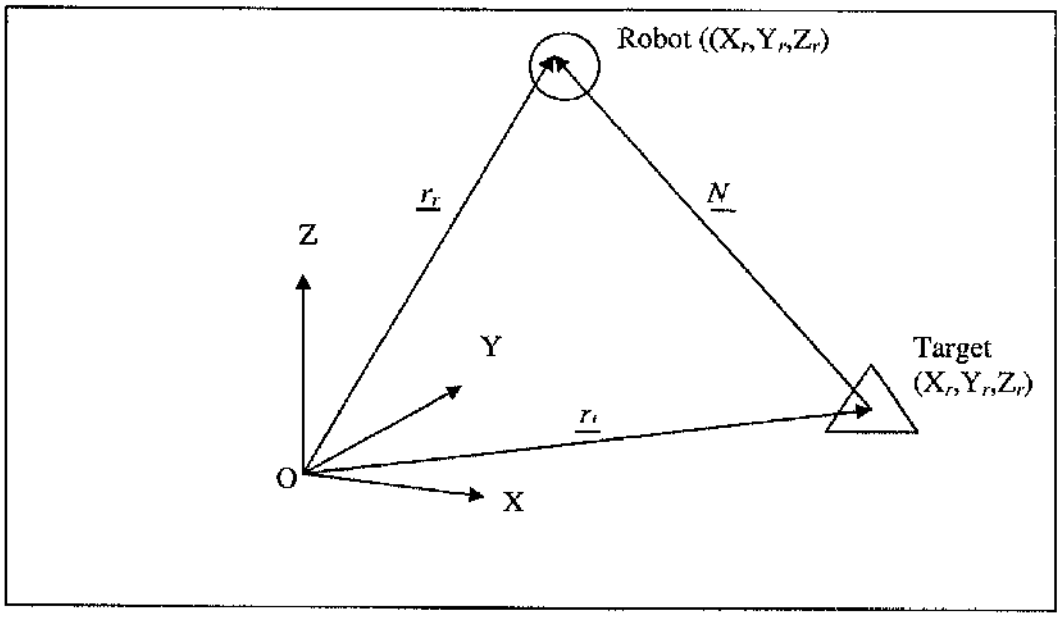


Figure 5.5: Unit normal between robot and target.

where:

$$N = \frac{(x_t - x_r, y_t - y_r, z_t - z_r)}{\left[(x_t - x_r)^2 + (y_t - y_r)^2 + (z_t - z_r)^2\right]^{1/2}} \tag{5.16}$$

From the transformation matrix developed in **Equation (5.8)**, a relationship is found which will point the camera along the unit normal towards the desired target direction:

$$\underline{n} = R \cdot \underline{N} \tag{5.17}$$

where $\underline{n}$ is the unit normal $\begin{bmatrix} n_x & n_y & n_z \end{bmatrix}$ expressed in body axes and R is the transformation matrix. Rearranging **Eq. (5.17)** gives:

$$\begin{bmatrix} N_x, & N_y, & N_z \end{bmatrix} = [R]^{-1} \begin{bmatrix} n_x, & n_y, & n_z \end{bmatrix} \tag{5.18}$$

Now, the transformation matrix belongs to the family of orthogonal matrices of which hold the properties:

"(1) *Real valued, non-singular matrices* [A] *are orthogonal if and only if* $[A][A] = [A][A^T] = [I]$ *(where* $[\ ]^T$ *indicates the transpose and [I] the identity matrix). It follows from this that* $[A]^{-1} = [A]^T$ *. If* [A] *is orthogonal, so are* $[A]^T$ *and* $[A]^{-1}$*. If* [A] *and* [B] *are orthogonal, the same is the case for the matrix product* [A][B].

(2) *The determinant of an orthogonal matrix equals either +1 or -1. If +1, the orthogonal matrices effect the transformation of a right-handed (left-handed) orthogonal co-ordinate system into another right-handed (left-handed) orthogonal system. Products of orthogonal matrices with determinant +1 are again of the same type.*" (Meyer, 1999)

From this, therefore, the inverse of the transformation matrix, $[R]^{-1}$ may be obtained by taking the transpose of $[R]$ such that:

$$[R]^{-1}\begin{bmatrix} n_x, & n_y, & n_z \end{bmatrix} = \begin{bmatrix} R^T \end{bmatrix}\begin{bmatrix} n_x, & n_y, & n_z \end{bmatrix} \tag{5.19}$$

where the transpose of the composite transformation matrix $\begin{bmatrix} R^T \end{bmatrix}$ may be found from the transpose of the individual transformation matrices:

$$\begin{bmatrix} R^T \end{bmatrix} = \begin{bmatrix} R_3^T \end{bmatrix}\begin{bmatrix} R_2^T \end{bmatrix}\begin{bmatrix} R_1^T \end{bmatrix} \tag{5.19b}$$

From **Equation (5.18)** the unit vector expressed in inertial axes, becomes:

$$\begin{bmatrix} N_x, & N_y, & N_z \end{bmatrix} = \begin{bmatrix} R^T \end{bmatrix}\begin{bmatrix} n_x, & n_y, & n_z \end{bmatrix} \tag{5.20}$$

where $\begin{bmatrix} R^T \end{bmatrix}$ can be solved from **Equation (5.5)**, **(5.6)**, **(5.8)** and **(5.19b)** and is found to be:

$$\left[R^T\right] = \left[R^{-1}\right] = \begin{bmatrix} c\theta_2 c\theta_3 & -c\theta_2 s\theta_3 & s\theta_2 \\ c\theta_3 s\theta_1 s\theta_2 + c\theta_1 s\theta_3 & c\theta_1 c\theta_3 - s\theta_1 s\theta_2 s\theta_3 & -c\theta_2 s\theta_1 \\ -c\theta_1 c\theta_3 s\theta_2 + s\theta_1 s\theta_3 & c\theta_3 s\theta_1 + c\theta_1 s\theta_2 s\theta_3 & c\theta_1 c\theta_2 \end{bmatrix} \quad (5.21)$$

Since the camera optical axis points along its y- body axis, as explained in **section 5.2**, then a unit vector pointing along the optical axis will have body-frame co-ordinates:

$$\left[n_x,\ \ n_y,\ \ n_z\right] = \left[0\ \ 1\ \ 0\right] \quad (5.22)$$

Therefore this reduces **Eq. (5.20)** whilst incorporating **Eq. (5.21)** to give an expression for a unit vector pointing along the camera optical axis expressed in an inertial co-ordinate reference frame as:

$$N_x = -\cos\theta_2 \sin\theta_3 \quad (5.23)$$

$$N_y = \cos\theta_1 \cos\theta_3 - \sin\theta_1 \sin\theta_2 \sin\theta_3 \quad (5.24)$$

$$N_z = \cos\theta_3 \sin\theta_1 + \cos\theta_1 \sin\theta_2 \sin\theta_3 \quad (5.25)$$

This may be simplified by observing from **figure 5.5** that there is no rotation of angle θ_2 around the y- body axis required to point the camera at the target if it is assumed the optical axis points along this axis. Therefore, by indeed pointing the camera optical axis along the y- body axis, **Equation (5.23) - (5.25)** reduce further, without loss of generality using $\theta_2 = 0$ to:

$$N_x = -\sin\theta_3 \quad (5.26)$$

$$N_y = \cos\theta_1 \cos\theta_3 \quad (5.27)$$

$$N_z = \cos\theta_3 \sin\theta_1 \quad (5.28)$$

By rearranging **Equation (5.26) - (5.28)** the camera orientation required to point towards the target expressed in Euler angles, given a unit normal between the camera and target, is found to be:

$$\theta_1 = \cos^{-1}\left(\frac{N_y}{\cos\left(\sin^{-1}\left(-N_x\right)\right)}\right) \tag{5.29}$$

$$\theta_2 = 0 \tag{5.30}$$

$$\theta_3 = \sin^{-1}\left(-N_x\right) \tag{5.31}$$

where $N_{x,y,z}$ are obtained from the target and camera positions using **Eq. (5.16)**.

5.4.1 Attitude Dynamics

To obtain a basic understanding of attitude control, knowledge of the rotational motion of the robot and the fundamentals of rigid body dynamics are required:

The term 'rigid body' refers to an object where its individual elements are fixed relative to each other i.e. as the body moves and rotates, the distance between the elements remains constant. A rigid body contains six degrees of freedom. Three of which refer to the translational motion of the centre of mass and the remainder to the rotational motion about the centre of mass. The following is a summary of the rotational motion of a rigid body, and for the purpose of this thesis, it is considered that the robot has a rigid body with fixed mass (Wertz, 1991), (Welsh, 1999).

5.4.2 Rotational Motion: The Euler Equations

For rotation of a rigid body in an inertial frame of reference, the total applied torque $\underline{M}$ equals the rate of change of angular momentum $\dot{\underline{H}}$:

$$\underline{M} = \dot{\underline{H}} \tag{5.32}$$

However, expressed in body axes, with angular velocity, $\underline{\omega}$ relative to an inertial frame, the total torque applied to the body is given as:

$$\underline{M} = \dot{\underline{H}} + \underline{\omega} \times \underline{H} \tag{5.33}$$

where $\underline{M}$ is now expressed in body axes.

Angular momentum $\underline{H}$ can be defined as:

$$\underline{H} = \underline{I} \cdot \underline{\omega} \tag{5.34}$$

where $\underline{I}$ is the inertia matrix.

If the body axes are chosen to be the principal axes, (i.e. those axes where the products of inertia are zero) the inertia matrix is most simply expressed as:

$$I = \begin{bmatrix} I_1 & 0 & 0 \\ 0 & I_2 & 0 \\ 0 & 0 & I_3 \end{bmatrix} \tag{5.35}$$

and therefore if I is a constant, assuming the robot is a completely rigid body, then:

$$\underline{M} = I \cdot \dot{\underline{\omega}} + \underline{\omega} \times (I \cdot \underline{\omega}) \tag{5.36}$$

Expressing the above **Equation (5.36)** in component form yields a set of three equations known as the *Euler Equations*, viz.

$$M_1 = I_1 \cdot \dot{\omega}_1 + (I_2 \cdot I_3)\omega_2\omega_3 \tag{5.37a}$$

$$M_2 = I_2 \cdot \dot{\omega}_2 + (I_3 \cdot I_1)\omega_3\omega_1 \tag{5.37b}$$

$$M_3 = I_3 \cdot \dot{\omega}_3 + (I_1 \cdot I_2)\omega_1\omega_2 \tag{5.37c}$$

The Euler Equations are non-linear, coupled first order differential equations and may be written in terms of $\dot{\underline{\omega}}$ as:

$$\dot{\omega}_1 = \frac{(I_3 - I_2)}{I_1}\omega_2\omega_3 - \frac{M_1}{I_1} \tag{5.38a}$$

$$\dot{\omega}_2 = \frac{(I_1 - I_3)}{I_2}\omega_1\omega_3 - \frac{M_2}{I_2} \tag{5.38b}$$

$$\dot{\omega}_3 = \frac{(I_2 - I_1)}{I_3}\omega_2\omega_1 - \frac{M_3}{I_3} \tag{5.38c}$$

Now that the Euler Equations have been defined, an attitude controller enabling camera pointing will be developed (Fortescue, 1991), (Chobotov, 1991), (Sidi, 1997), (Thomson, 1986).

5.5 Developing the Attitude Control System

Camera fixed pointing for space applications requires a closed-loop control that combines full autonomy with minimal hardware, due to space restrictions within the free-flyer, and limited software, with space qualified processor technology lagging in comparison to terrestrial advances.

Ensuring camera fixed pointing requires the modification of parameters governing camera orientation; as well as keeping these parameters at fixed values regardless of the disturbances. These modifications are performed using a feedback controller which work based on error control (Glenmar, 1999), (McQuade, 1997). With this principle, the parameter to be fixed is continually compared to a reference signal, and the resulting error amplified in order to drive the system actuators. The control aims to eliminate the error regardless of the disturbances acting on the system and maintain stability.

There is a methodology that meets these specifications and will be discussed, known as *Lyapunov's Second Method* - well known and now extensively applied to complex space applications (Rouche, Habets & Laloy, 1977).

5.5.1 Lyanpunov's Method

Lyanpunov observes and exploits a deceptively simple idea to devise a function that will drive a system to stablility, such that:

'A dynamical system is stable at an equilibrium state x_e (in the sense that it returns to equilibrium after any perturbation) if and only if there exists a 'Lyapunov function,' i.e. some scalar function V(**x**) *of the state with the properties:*

$$i) \qquad V(\underline{x}) > 0,\ \dot{V}(\underline{x}) < 0 \quad \text{when } x \neq x_e \tag{5.39}$$

and

$$ii) \qquad V(\underline{x}) = \dot{V}(\underline{x}) = 0 \qquad \text{when } x = x_e \tag{5.40}$$

The Lyapunov function, whose properties are expressed above, ensures the state vector converges at the global minimum of the function. Convergence is controlled by the rate of change of the function by ensuring $\dot{V}(\underline{x})$ is always negative definite. If $\dot{V}(\underline{x})$ becomes positive, control intervention is required to inhibit the divergence of the state vector from the goal point. When the function and the rate of change of potential function are zero, the goal has been reached. Kalman and Bertam describe the notion of a Lyapunov function, sometimes termed a potential function, as:

'If the rate of change dE(x)/dt of the energy E(x) of an isolated physical system is negative for every possible state **x**, except for a single equilibrium state x_e, then the energy will continually decrease until it finally assumes its minimum value $E(x_e)$.' (Kalman, 1960)

5.5.2 *Achieving Convergence*

Since an analytical definition of the rate of change of a potential function $\dot{V}(\underline{x})$ based on the state vector (x) can be obtained, viz:

$$V = f(\mathrm{x}) \tag{5.41}$$

$$\Rightarrow \quad \dot{V} = \nabla f . \dot{\mathrm{x}} \tag{5.42}$$

it then becomes possible to calculate the requirements needed to force the vector to converge on a goal point by ensuring the control continually maintains $\dot{V}(\underline{x}) < 0$.

5.5.3 *Potential Function Derivation*

To bring the camera to rest in some desired direction requires control of both the camera's Euler angles and its body rates. Therefore, a Lyapunov function may be defined as:

$$\mathrm{V} = \mathrm{V}_{\mathrm{Euler}} + \mathrm{V}_{\mathrm{body\ rates}} \tag{5.43}$$

The Euler function is expressed as a quadratic with a single goal point ensuring the potential $\mathrm{V}_{\mathrm{Euler}}$ is positive for every orientation except at the solution, where $\left(\theta_i - \theta_i^*\right)$ is zero:

$$\mathrm{V}_{\mathrm{Euler}} = \frac{k}{2}\sum_{i=1}^{3}\left(\theta_i - \theta_i^*\right)^2 \tag{5.44}$$

where θ_i^* is the desired orientation as defined in **Equations (5.29) - (5.31)**, θ_i is the actual orientation of the robot, which was set and known by the operator, and input into the control system, and k is a scaling constant. For the body rate potential, the goal will be attained when the body rates are zero, so that it has a functional form:

$$V_{\text{body rates}} = \frac{1}{2}\sum_{i=1}^{3} I_i \cdot \omega_i^2 \tag{5.45}$$

Combining both the Euler potential and the body rate potential defines the total Lyapunov function as:

$$V = \frac{k}{2}\sum_{i=1}^{3}\left(\theta_i - \theta_i^*\right)^2 + \frac{1}{2}\sum_{i=1}^{3} I_i \cdot \omega_i^2 \tag{5.46}$$

which, to conform to Lyapunov's Theorem, must maintain $\dot{V}$ negative definite at every time-step. Differentiating **Equation (5.46)** to give $\dot{V}$:

$$\dot{V} = I_1\omega_1\dot{\omega}_1 + I_2\omega_2\dot{\omega}_2 + I_3\omega_3\dot{\omega}_3 + k\left(\left(\theta_1 - \theta_1^*\right)\dot{\theta}_1 + \left(\theta_2 - \theta_2^*\right)\dot{\theta}_2 + \left(\theta_3 - \theta_3^*\right)\dot{\theta}_3\right) \tag{5.47}$$

And substituting the Euler Equations expressed in **Equations 5.37(a)** through **(c)** into **(5.47)**, it is found that:

$$\dot{V} = \omega_1 M_1 + \omega_2 M_2 + \omega_3 M_3 + k\left(\left(\theta_1 - \theta_1^*\right)\dot{\theta}_1 + \left(\theta_2 - \theta_2^*\right)\dot{\theta}_2 + \left(\theta_3 - \theta_3^*\right)\dot{\theta}_3\right) \tag{5.48}$$

Further, substituting **Equations (5.12)** through **(5.14)** expands the function, viz:

$$\begin{aligned}\dot{V} = {} & \omega_1 M_1 + \omega_2 M_2 + \omega_3 M_3 + k\left(\theta_1 - \theta_1^*\right)\left(\frac{\omega_2 \sin(\theta_3) + \omega_3 \cos(\theta_3)}{\cos(\theta_2)}\right) \\ & + k\left(\theta_2 - \theta_2^*\right)\left(\omega_2 \cos(\theta_3) - \omega_3 \sin(\theta_3)\right) \\ & + k\left(\theta_3 - \theta_3^*\right)\left(\omega_1 + \left(\omega_2 \sin(\theta_3) + \omega_3 \cos(\theta_3)\right)\tan(\theta_2)\right)\end{aligned} \tag{5.49}$$

Now, on analysis of the above equation, and in keeping with Lyapunov's Theorem, it may be found that the condition $\dot{V} < 0$ will be satisfied if:

$$M_1 = -k\left(\theta_3 - \theta_3^*\right) - k\omega_1 \tag{5.50}$$

$$M_2 = -k\left(\theta_3 - \theta_3^*\right)\sin\theta_3 \tan\theta_2 - k\left(\theta_2 - \theta_2^*\right)\cos\theta_3 - k\left(\theta_1 - \theta_1^*\right)\sin\theta_3 \sec\theta_2 - k\omega_2 \tag{5.51}$$

$$M_3 = -k\left(\theta_3 - \theta_3^*\right)\cos\theta_3 \tan\theta_2 - k\left(\theta_2 - \theta_2^*\right)\sin\theta_3 - k\left(\theta_1 - \theta_1^*\right)\cos\theta_3 \sec\theta_2 - k\omega_3 \tag{5.52}$$

Such that:

$$\begin{aligned}
\dot{V} &= \omega_1(-k\left(\theta_3 - \theta_3^*\right) - k\omega_1) \\
&+\omega_2(-k\left(\theta_3 - \theta_3^*\right)\sin\theta_3 \tan\theta_2 - k\left(\theta_2 - \theta_2^*\right)\cos\theta_3 - k\left(\theta_1 - \theta_1^*\right)\sin\theta_3 \sec\theta_2 - k\omega_2) \\
&+\omega_3(-k\left(\theta_3 - \theta_3^*\right)\cos\theta_3 \tan\theta_2 - k\left(\theta_2 - \theta_2^*\right)\sin\theta_3 - k\left(\theta_1 - \theta_1^*\right)\cos\theta_3 \sec\theta_2 - k\omega_3) \\
&+k\left(\theta_1 - \theta_1^*\right)\left(\frac{\omega_2 \sin(\theta_3) + \omega_3 \cos(\theta_3)}{\cos(\theta_2)}\right) \\
&+k\left(\theta_2 - \theta_2^*\right)\left(\omega_2 \cos(\theta_3) - \omega_3 \sin(\theta_3)\right) \\
&+k\left(\theta_3 - \theta_3^*\right)\left(\omega_1 + \left(\omega_2 \sin(\theta_3) + \omega_3 \cos(\theta_3)\right)\tan(\theta_2)\right)
\end{aligned} \tag{5.53}$$

while k>0. From this expression, it can be seen that as $\theta_i \to \theta_i^*$ (i = 1-3), this reduces to:

$$\dot{V} = -k\omega_1^2 - k\omega_2^2 - k\omega_3^2 \tag{5.53}$$

which is always negative definite, and such that the camera will always come to rest at the desired orientation.

These equations are incorporated into the control law, producing a feedback loop that processes the orientation angles, comparing them with the desired

orientation. An error signal is produced $\theta_e = \sum_{i=1}^{3}\left(\theta_i - \theta_i^*\right)$ which diminishes to zero as the desired orientation is reached. Note the control is non-linear and ensures global convergence to the desired attitude by continually controlling the rate of change of potential.

From **chapter 2** the characteristics of the robot were defined as a sphere of mass m_r of 2.5kg and radius r_r 0.15 metres. The moment of inertia is thus found to be:

$$I_{1,2,3} = \frac{2}{5} m_r r_r^2 = 2.5 \text{ kgm}^2 \tag{5.54}$$

The robot has an initial orientation of $\theta_1 = \theta_2 = \theta_3 = 0$ radians, set and known by the operator, and input into the control system, and a desired orientation was chosen as $\theta_1 = -0.7$ rads, $\theta_2 = 0$ rads, $\theta_3 = 0.7$ rads, for evaluation. The simulation was run for a period of 50 seconds, during which time the control law feedback loop compared the desired orientation with the actual to produce an error signal. The error signal and hence the Lyapunov function was reduced to zero ensuring the rate of change of Lyapunov function negative definite.

Figure 5.6 demonstrates the smooth re-orientation of the camera from its initial attitude to the specified attitude over the stated time period given these initial conditions. This demonstrates the strong and immediate influence of the control system governed by the potential function, which drives the camera to its required orientation. The robot's position is governed independently by Braitenberg laws controlled by sensory information, and is not affected by the camera control torque in any way. In **Figure 5.7** the corresponding body rates over time are shown, which diminish to zero when the desired robot orientation is reached, as expected. In addition, the torque activity is illustrated in **figure 5.8**, which clearly decays to zero as the goal orientation is approached. The coupling between the three axes is evident by observing that T_2 is displaced as a consequence of the control on T_1 and T_3.

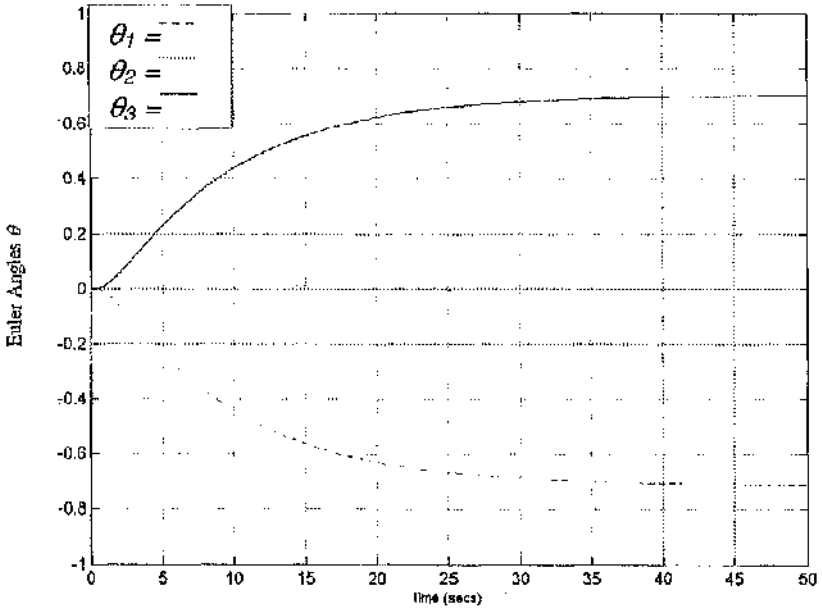


Figure 5.6: Euler angles control.

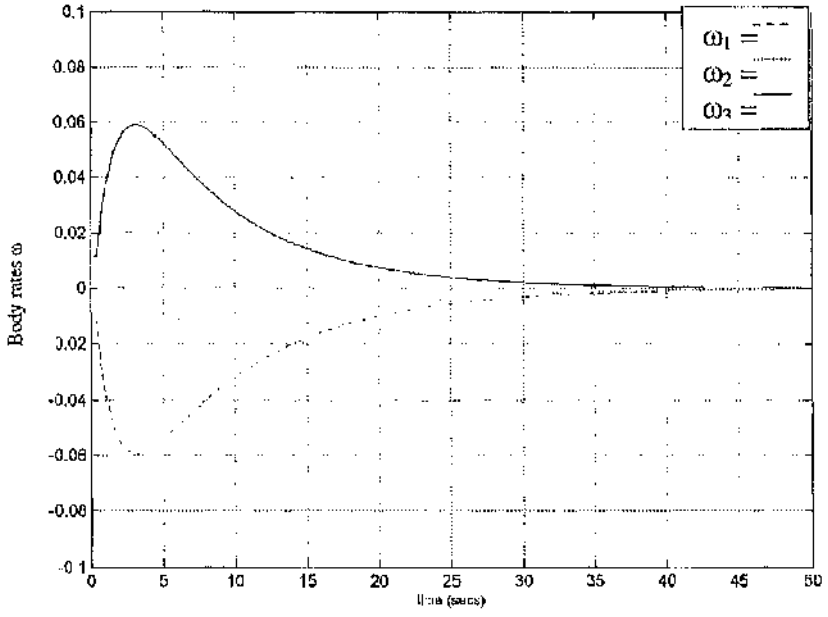


Figure 5.7: Body rates over time.

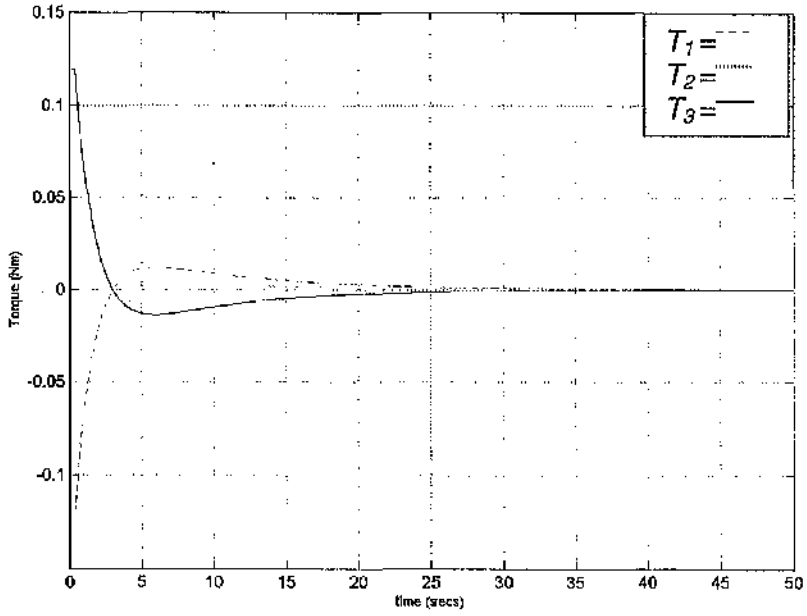


Figure 5.8: Control torque stability over time.

Figures 5.9 and **5.10** illustrate the potential function activity and the rate of change of potential function respectively over the same time period. Conforming to Lyapunov's Theorem the graphs depict a positive definite potential function which reduces to zero as the orientation converges on its goal, and, in addition, the rate of change of potential function which is negative definite and reduces to zero.

These results indicate the success of this method on controlling the camera orientation. **Figure 5.12** shows a series of snapshots of the camera re-orientating from the initial attitude to the specified target attitude for the simulation run above. The view is taken from virtual camera lens where the green triangular pyramid in each still represents the target orientation desired. (**Figure 5.11** shows the external view of the robot and target, to ease interpretation of **figure 5.12**). As can be seen from the still frames the camera slews from its original orientation to come to rest viewing the goal point.

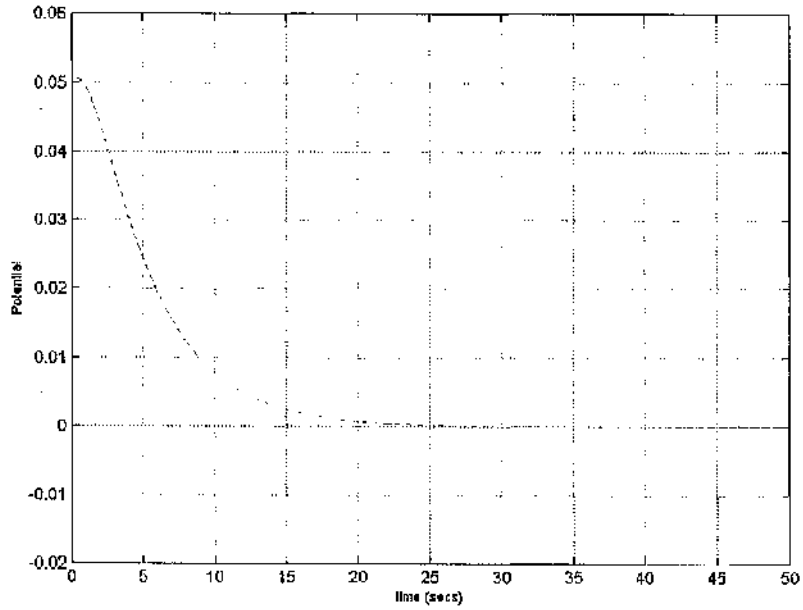


Figure 5.9: Potential function positive definite.

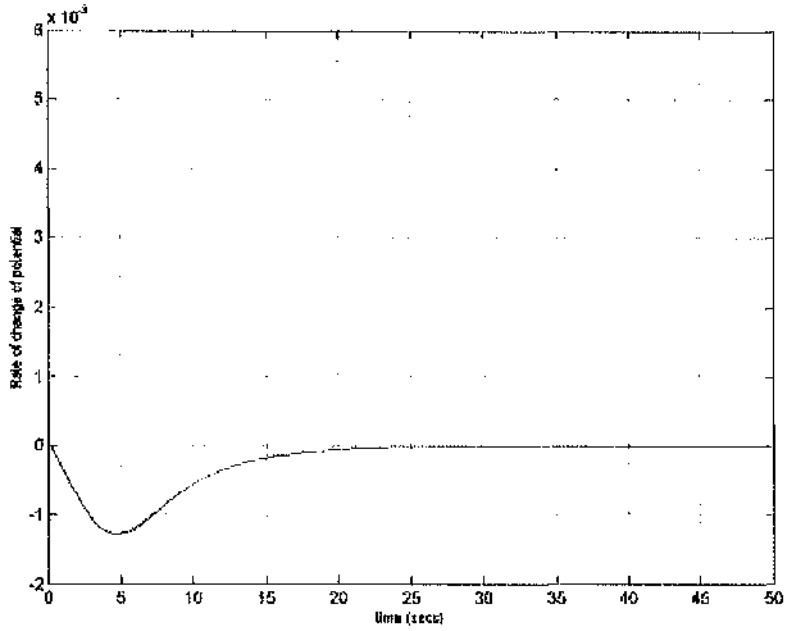


Figure 5.10: Rate of change of potential function negative definite.

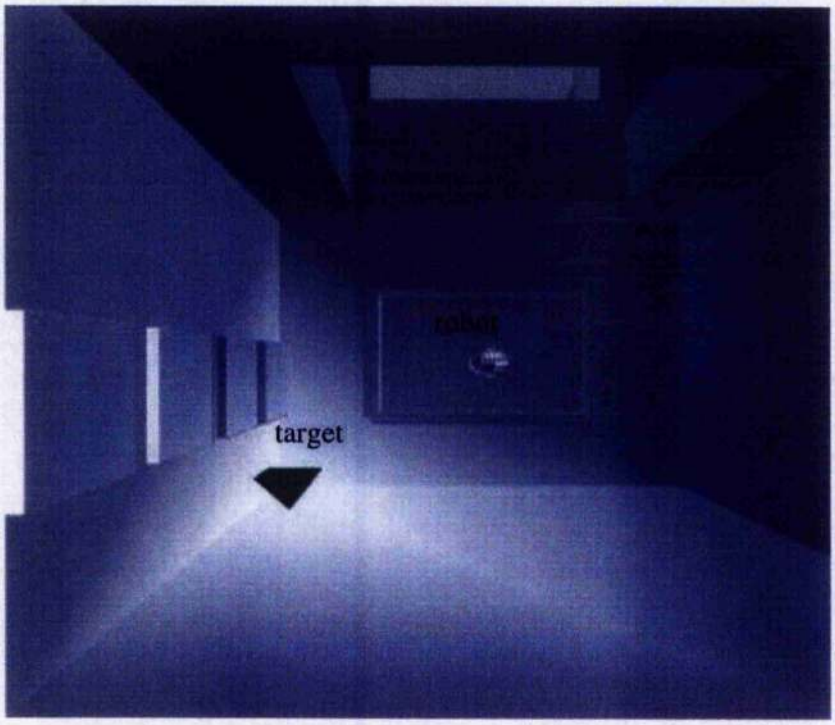


Figure 5.11: External view of robot and goal.

5.6 Conclusion

It was assumed a camera tool is fitted to the robot, and from this assumption, the chapter began by successfully developing the virtual view seen from the camera tool, providing an initial camera orientation.

The chapter then developed an attitude control system using Lyapunov's method to bring the camera optical axis to rest at the desired orientation. From the results given, it is shown that Lyapunov's method successfully enables stable convergence to the desired orientation configuration with minimal complexity, by the simple derivation of a potential function.

This now provides the free-flyer with a useful function as a camera aide, able to adopt a desired orientation autonomously whilst avoiding collision hazards.

In the following chapter, the control system returns to Braitenberg control methodologies and *klinokinesis* methods to enable the camera controller to reach (or follow) a moving goal. This is used alongside the Euler control law developed independently in this chapter, to allow orientation towards the target whilst following it.

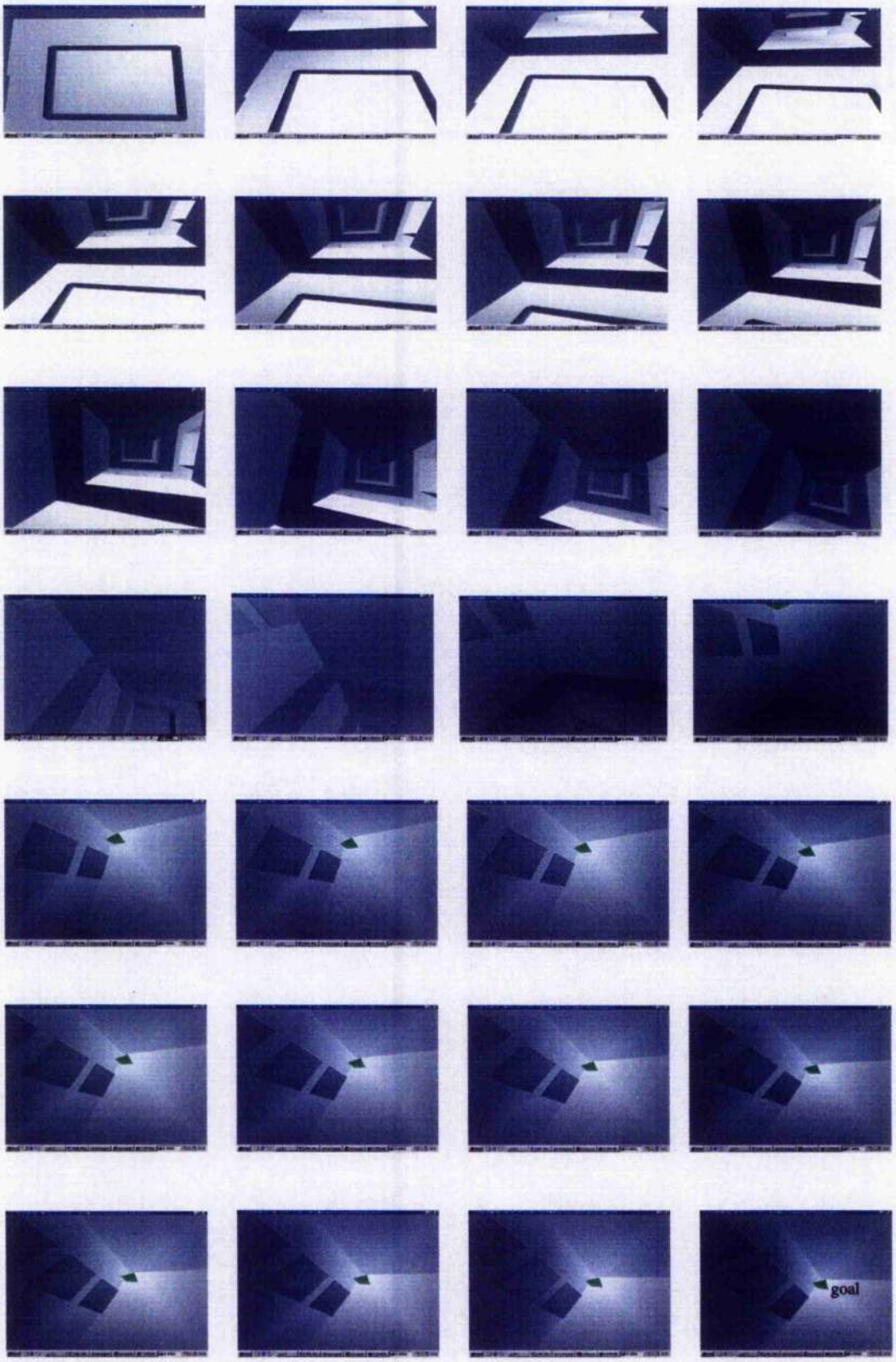


Figure 5.12: Unit normal between robot and target.

Chapter Six: Goal Strategies

'Curse my metal bones!'

C_3PO

6.1 Introduction

The previous three chapters have seen the systematic evolution of a working control methodology for implementation within a free flying robotic vehicle. The vehicle is fully autonomous and manoeuvres, collision free, in the confines of a 3-dimensional workspace environment with the capability of camera fixed pointing.

In this chapter, the task of goal searching is introduced, where guidance strategies are implemented in order to plan and execute a path between an initial start point and a final goal destination. The methods examined are dependent on the emission of a measurable signal (thermal, olfactory, acoustic or likewise) whereby a signal strength gradient then becomes the means of selecting an efficient path and hence locating the signal source. Two methods exploited by certain biological systems to navigate to their desirable goal, *klinokinesis* and *tropotaxis*, (Gillies, McInnes & Neil 1994) are introduced and their merits discussed. The method of klinokinesis is adopted and integrated into the control methodology, while the goal destination is simulated to emit a chemical signal dispersed into the enclosed environment. Thus the robot can follow the chemical gradient to reach the goal, the applications of which are discussed below.

Incorporating goal strategies to the control system enhances the robot's functions, and, by varying the sensor type fitted, shall vary the robot's capabilities. The robot may be used to investigate and detect the source of a potentially hazardous gas leak, or fire, in a sealed off module using chemical or thermal sensors in view of maintaining crew safety. Or, the robot may use the goal-searching algorithm to return

to home base through a photosensitive or acoustic recall method. Or finally, it may use infrared detection sensors to detect and follow an astronaut, whilst awaiting further commands, or for video conferencing. In each case, the robot functions and capabilities are greatly enhanced by the incorporation of this algorithm.

6.2 Guidance Strategies

Developing guidance strategies to reach a goal destination point is a frequently encountered problem. It is possible, however, to analyse and mimic the strategies employed by simple biological organisms such as the methods of *klinokinesis* and *tropotaxis* (Menzies, Das & Wood, 2006), (Bell & Carde 1984). These methods depend on measuring a signal emitted from the goal. The signal may be acoustic, chemical or thermal so long as there is a signal concentration gradient between the start point and goal destination. These methods, adopted for robotics, make a useful study for the purposes of this investigation since they require low computational power. This is because no forward planning or pre-knowledge of the environment is required, the path chosen to reach the goal destination is determined purely from the environmental cue provided by the path of the concentration gradient.

Klinokinesis is the simpler of the two methods and involves measuring the concentration of the signal at one point, then moving to the next spatial point and measuring again. If the temporal change in signal intensity is favourable, the robot continues in this direction, measuring at each time step. If, however, the temporal change in concentration at any time becomes unfavourable, the robot changes its direction of motion by some random rotation. The strategy is referred to as indirect guiding, as the directional changes are not influenced by the gradient orientation, but are purely random (Wilson 1991), (Gillies, 1994). The possibility of encountering local minima is reduced by the incorporation of the random variable.

Tropotaxis involves measuring the signal strength at more than one sensor (often bilateral), placed symmetrically. The goal destination is reached by orienting the robot such that the signal intensity measured from the symmetrical sensors are equal, and so ensuring the direction of motion is along the concentration gradient.

Tropotaxis has the advantage of enhancing path planning, since the robot determines the chemical gradient *before* movement and since obstacles block the signal, then the robots orientated direction would never occur heading towards an obstacle.

However, since the distance between the symmetrical sensors is constrained by the size of the robot, then tropotaxis is only useful when the concentration gradient is steep, or when the sensors are highly sensitive. For this reason, the method of klinokinesis is investigated here.

6.3 Klinokinesis

In using the klinokinesis strategy, the robot measures the concentration of the signal at one point. If no detectable signal can be measured, such that the signal intensity falls below the threshold value detectable by the sensors, then the robot changes direction by some arbitrary rotation and moves forward. It does so until a signal can be detected. When an identifiable signal can be detected, the robot moves forward and measures again. If the temporal change in signal intensity is favourable the robot takes a new step in this direction. However, if the change in signal intensity at any point becomes unfavourable, the robot, again, changes direction by some random amount. By continuing in this manner, the source of the signal is reached successfully using little computational power.

The dispersion of chemical signals in still air may be approximated by using the chemical diffusion equation:

$$\frac{\partial C_{signal}}{\partial t} = D\nabla^2 C_{signal} \tag{6.1}$$

where C_{signal} is the chemical concentration and D is the binary molecular diffusion coefficient, incorporating distributed noise arising from turbulent eddies, and locomotion errors.

The concentration gradient is calibrated inversely proportional to the distance from the goal centre such that there is an inverse concentration gradient of maximum value at the source and reducing linearly outward.

The co-ordinates of the goal point and the co-ordinates of the start point are defined, and as such the distance from the goal is then calculated as:

$$r = \sqrt{\left(x_{start} - x_{goal}\right)^2 + \left(y_{start} - y_{goal}\right)^2 + \left(z_{start} - z_{goal}\right)^2} \tag{6.2}$$

In a steady state, it is assumed that:

$$\frac{\partial C_{signal}}{\partial t} = 0 \tag{6.3}$$

such that:

$$\frac{\partial^2 C_{signal}}{\partial^2 t} = 0 \tag{6.4}$$

a solution to (6.1) is:

$$C_{signal} = \frac{Q}{r} \tag{6.5}$$

where Q is a calibration constant, here chosen to be 100, selected on the basis of the environment dimensions and with the intention of magnifying the concentration to a clear value.

Given the concentration levels at any point, a filter was included to represent the levels of concentration that would be too low for the sensor to detect. This was chosen to be those levels at a distance of 15 metres on x, y and z-axes, which would provide a C_{signal} value of around 4 units. This was chosen to allow the robot to detect a signal concentration from any point within the one module of whose dimensions are

15 metres. Further, the robot was instructed to stop just before the goal point was reached (to avoid colliding with the goal) by commanding that acceleration and velocity be reduced to zero whenever the signal concentration reached a predetermined value. In this case chosen to be 100 such that the robot would stop at a distance of 1 metre from the goal. Varying the concentration level at which the robot is commanded to stop, will automatically vary the distance from the goal at which the robot will come to rest. Again, we are envisaging that the robot is interfaced to a chemical sensor for gas leak or fire detection applications.

With the signal concentration at every point established, a route to the goal position is achieved by using a random number generator to generate 'goal searching' components of acceleration. Progress to the next point through integration of the acceleration components proceeds and the concentration level is recalculated. If the measured concentration is found to be greater than or equal to the previous recorded concentration, then the 'goal searching' components of acceleration remain as for the previous step. If the measured concentration is found to be less than the previous recorded concentration, then new acceleration components are generated from the random number generator.

6.4 The Braitenberg Structure

Figure 6.1 illustrates the structure of the Braitenberg control system complete with obstacle avoidance, goal seeking, and speed control algorithms, camera control torque and an acceleration controller. Sensors take information on the range to the closest obstacle along the sensor line-of-sight and within its horizon, combined with information gathered on the goal signal concentration levels and safety speed levels to drive the control system. Within the controller, the behaviours are weighted and summed and the output is directed to the thrusters, which manifest the desired behaviour.

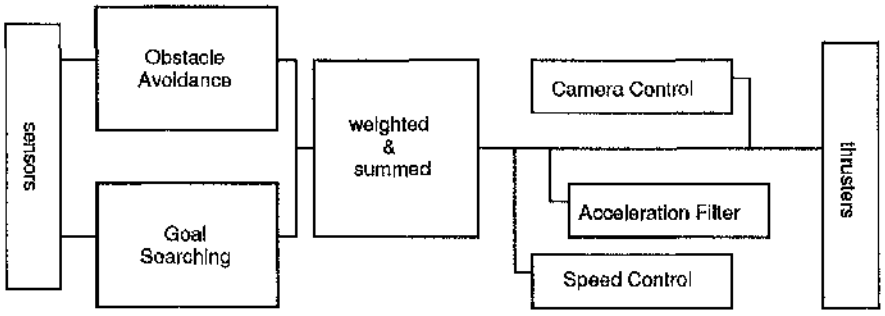


Figure 6.1 Control system structure featuring obstacle avoidance and goal searching.

Later, it will be shown that the total acceleration, $\underline{a}_{total}$, is calculated from the sum of the weighted components of acceleration arising from each behaviour, and whose individual weightings will become variable, dependent on vehicle state. This allows a smooth transition between one behaviour and the next as the vehicle encounters a new situation.

6.5 Testing the Model

Figure 6.2 plots the path taken by the robot from an initial start point with co-ordinates (3, 3, 3) to reach goal points with co-ordinates (11, 3, 4). In **figure 6.3**, the goal point has co-ordinates (9, 8, 4), **figure 6.4** has goal position (11, 11, 8) and in **figure 6.5**, the goal point is given as (7, 8, 19). In each case, the signal is distributed throughout the environment with signal intensity greatest at the goal point and which diminishes with distance, and from this, the goal point is found by signal sensing. The paths taken illustrate the obstacle avoidance algorithm in addition to the goal finding capabilities of the model.

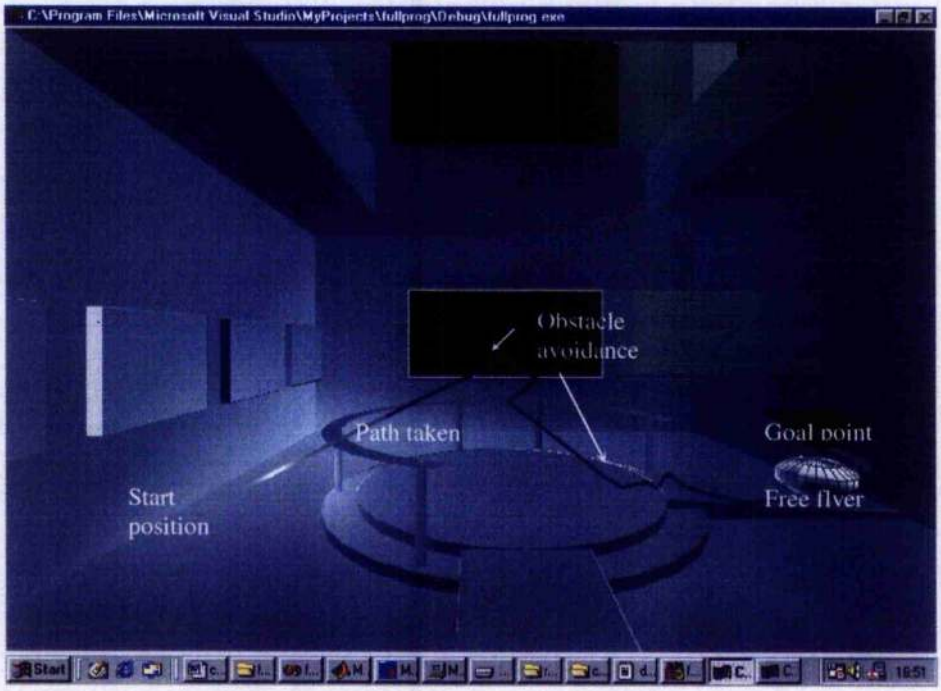


Figure 6.2 Goal searching behaviour

Figure 6.3 Goal searching behaviour

Figure 6.4 Goal searching behaviour

Figure 6.5 Goal searching behaviour

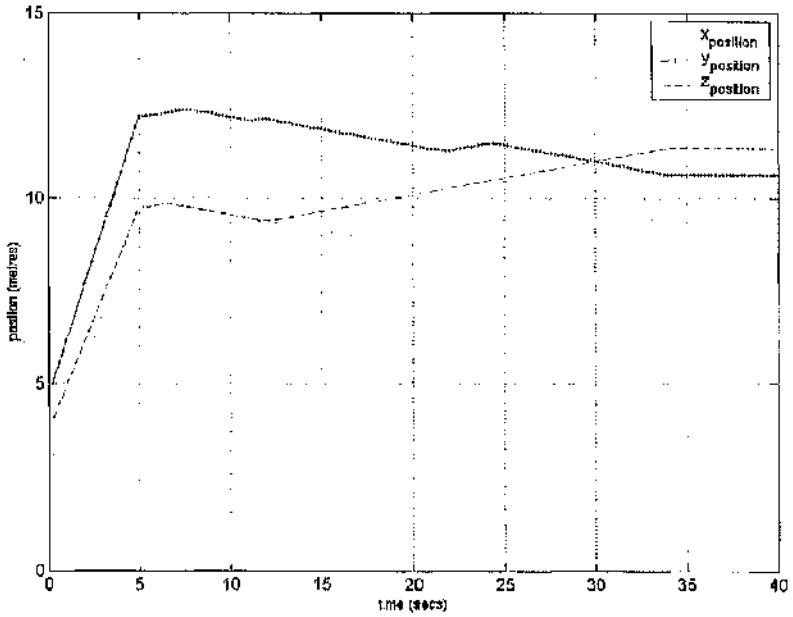


Figure 6.6 Graph plotting the x, y, z co-ordinates of the robot position against time as it tracks the goal point.

Figure 6.6 plots the x, y, z co-ordinates of the path taken by the robot against time as it traces the goal point. The robot had start co-ordinates (3, 5, 4) and goal co-ordinates (10.5, 10.5, 10.5). The plot shows the robot successfully reaches its destination by coming to rest a short distance from the goal to avoid collision.

6.6 Moving Goal Tracking

With the goal strategy proving successful, it was then tested on a moving goal such that it may incorporate a useful function to enable it to follow an astronaut awaiting further commands, or for video conferencing; or to function as a tracker for any other desired moving goal point.

For this test, the moving goal point was simulated as a floating solid sphere, (to contrast with the mesh sphere representing the free-flying robot) and is given an initial start position, as chosen by the operator. The moving goal was then given its own initial speed and direction, and programmed to have its own obstacle avoidance

capabilities, speed controller and acceleration filter. The goal was then allowed to manoeuvre randomly around in the environment avoiding collisions. Further, the maximum velocity of the free-flying goal was chosen as 0.09 ms^{-1}, slightly less than the maximum velocity for the free-flying robot. This was to allow the robot to catch up with the goal. The robot was given its initial start position and, using the same strategy as for the fixed goal searching, was commanded to track the moving goal.

Figure 6.7 shows a series of still frames, which demonstrate moving goal tracking. The initial conditions for the moving goal were:

Initial speed and direction:	$v_x = 0.07\ ms^{-1}$
	$v_y = 0.04\ ms^{-1}$
	$v_z = 0.02\ ms^{-1}$
Maximum velocity:	$v_{maximum} = 0.09\ ms^{-1}$
Start co-ordinates:	(6, 5, 4)

And for the free-flying robot, initial conditions were given as:

Initial speed and direction:	$v_x = 0.04\ ms^{-1}$
	$v_y = 0.08\ ms^{-1}$
	$v_z = 0.03\ ms^{-1}$
Maximum velocity:	$v_{maximum} = 0.1\ ms^{-1}$
Start co-ordinates:	(3, 3, 3)

From the screenshots, the free-flying robot is represented as a mesh sphere, and the moving goal as a solid sphere. In addition, in the top left hand corner of each screenshot is a view from the camera, to show that the camera-pointing tool is likewise capable of tracking a *moving* goal, where the sphere silhouette in each shot represents the goal target. It is clear from the screenshots that the moving goal tracking capabilities of the model are successful, as is moving goal camera tracking.

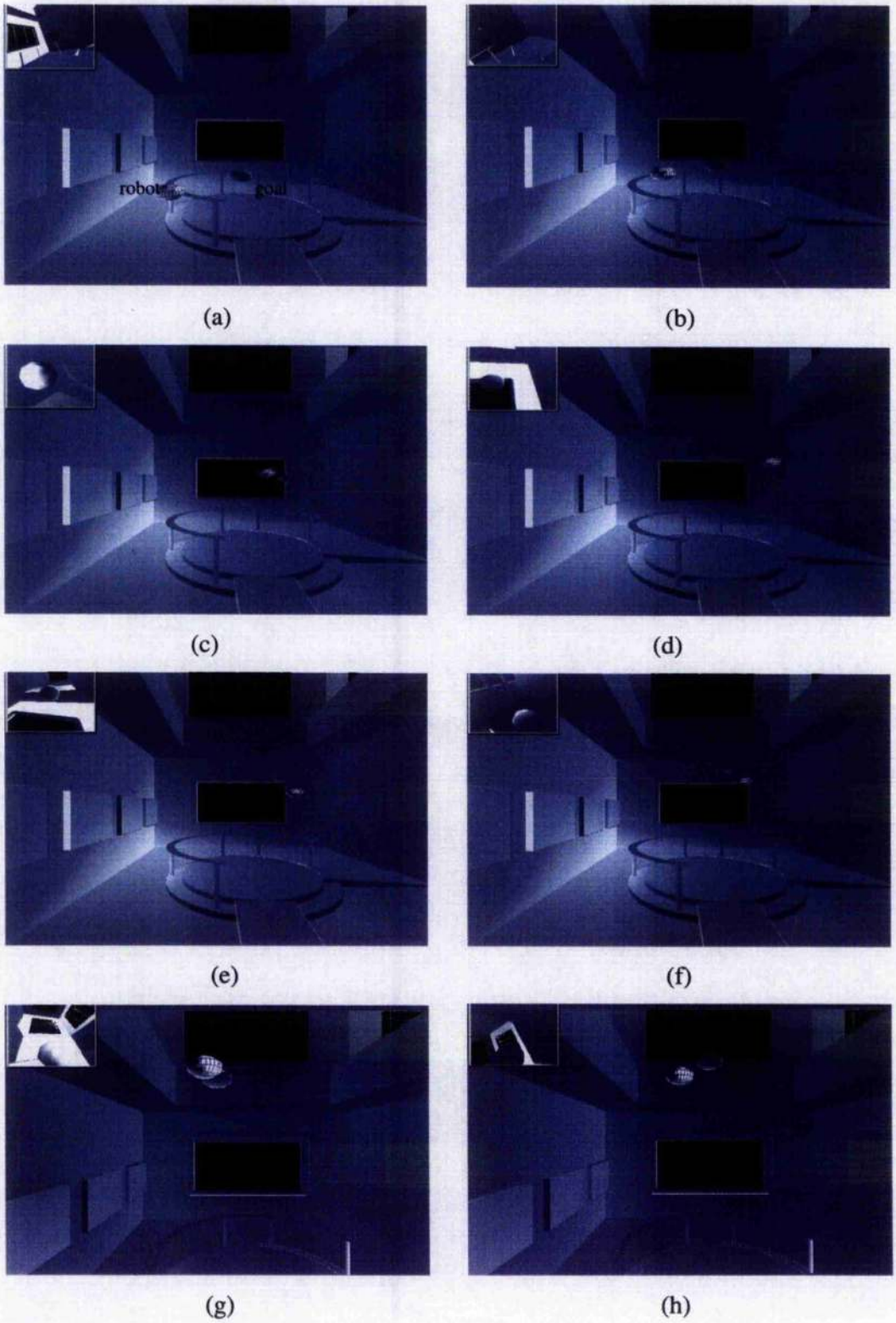


Figure 6.7 (a) - (p): Demonstrating the moving goal tracking and camera pointing capabilities of the model.

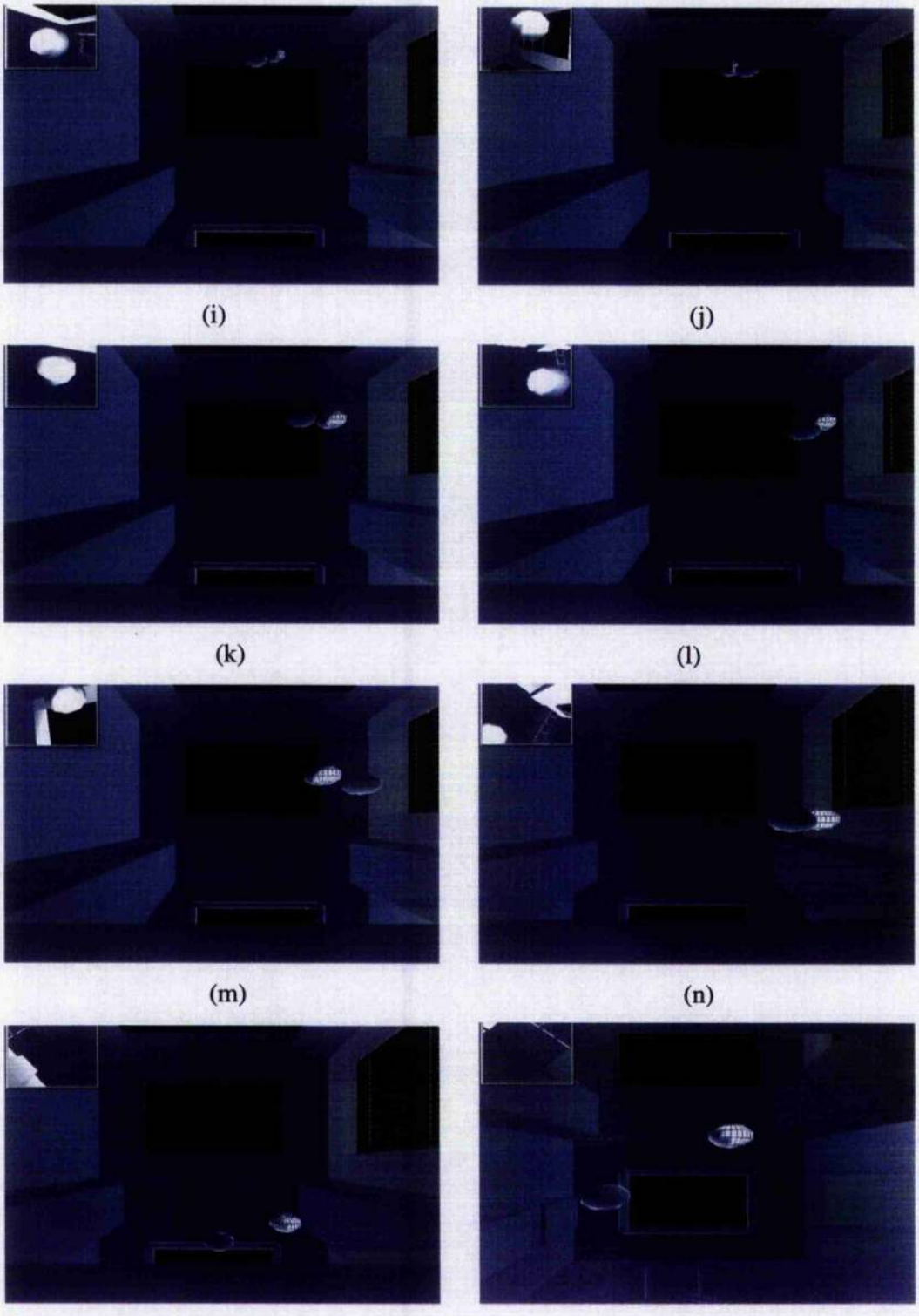


Figure 6.7 (a)-(p): Demonstrating the moving goal tracking and camera pointing capabilities of the model.

This is a really nice demo of the method employed here to track moving goals. The results are further highlighted in **figure 6.8**, which superimposes the x, y, z components of the free-flying robot onto the components of the path of the moving goal.

From the following trace, the success of the robot to track a moving goal is clear, with the robot closely shadowing the goal at all times during the 900 second run.

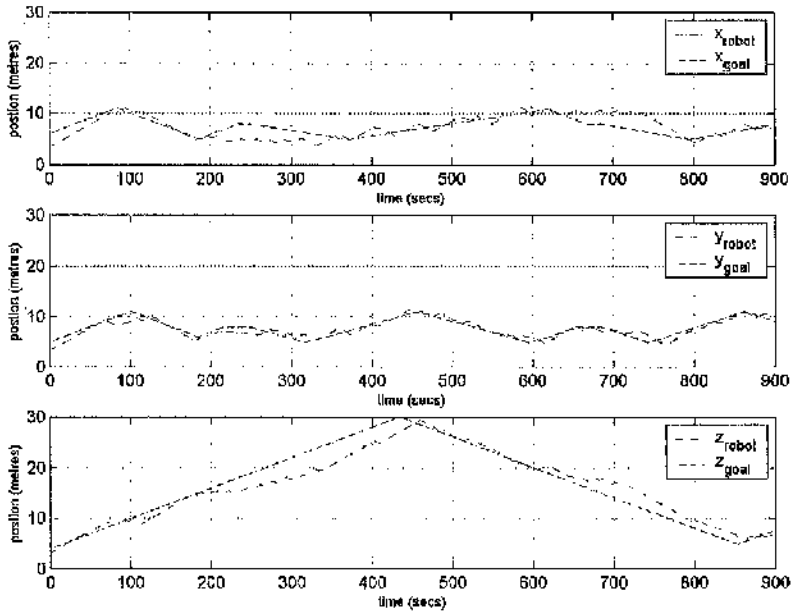


Figure 6.8: Trace of the moving goal path and robot path to demonstrate goal-tracking capabilities.

6.7 Discussion

A guidance strategy was implemented to devise a path between an initial start point and a final goal destination. The technique employed was based on the method of klinokinesis, one exploited by certain biological systems to navigate to their desirable goal by measuring a signal emitted from the goal. The signal strength gradient then becomes the means of selecting an efficient path.

The signal strength gradient was calibrated inversely proportional to the distance from the goal centre such that there was a linear concentration gradient of maximum value at the source and reducing linearly outward. Then, by methods of klinokinesis, the robot is shown to navigate successfully towards its goal destination, coming to rest at a distance from the goal.

The method was tested tracking a moving goal point, where the moving goal would represent an astronaut or another free-flyer, complete with its own obstacle avoidance algorithm, speed controller and acceleration filters. The results were shown to be successful, with consistent tracking of the moving goal. In the previous chapter, an Euler control law was developed independently to allow orientation towards the target, whilst this chapter's methodologies of klinokinesis independently allow following it. Whilst each of these methodologies are independent, they are integrated together to allow a versatile tracking system.

The goal searching behaviour was weighted alongside the collision avoidance algorithm and incorporated into the control system, integrated with the camera control torque, acceleration controllers and speed controllers, providing an integrated control system with a viable function.

Chapter Seven: Self Sufficiency

'Daisy, Daisy,
Give me your answer, do'

Hal
2001- A Space Odyssey

7.1 Introduction

The previous chapters have followed the development of a robot model complete with useful functionality for operation in space. However, to be truly autonomous, the robot must display a level of self-sufficiency, which will ensure it never runs out of fuel or places itself in an irrecoverable position over a long period of time whilst carrying out mission tasks.

The need to carry out these mission tasks whilst maintaining functionality provides the robot with conflicting motivations that must be addressed to ensure the continuity of an efficient useful system.

This chapter addresses the problem of self sufficiency, by proposing a functional method based on a basic cycle of: work - finding fuel - refuelling, and through the implementation of a control mechanism based entirely upon motivational tendencies, known as the *cue-deficit* model, an efficient control architecture is developed, allowing decisions to be made on the robot's best use of resources and time.

7.2 Self-Sufficiency and the Two Resource Problem

Self-sufficiency remains a fundamentally important aspect of autonomy. To be a self-sufficient an agent, just like biological systems, necessitates the ability to sustain itself over an extended period of time. This implies that the agent never finds itself in any irrecoverable position or with any irrecoverable deficiency in any vital resource necessary to perform its design function. In addition to this behavioural stability, a robot must have market viability, such that it satisfies its employer by being both behaviourally stable whilst able to perform the tasks it was designed for. This is a fundamental constraint for the success of a long-lived autonomous agent. In this design case: the robot must never run out of fuel, or find itself in a position that will inevitably lead to running out of fuel (McFarland and Spier, 1997).

The minimum requirement for a useful autonomous agent, is for it to be able to sustain itself whilst performing the design tasks required of it, such as searching for a goal point, collecting data, refuelling etc. Sequencing these possibly conflicting tasks to ensure the agent never 'dies' by refuelling appropriately, whilst preventing the size of its workload from persistently growing, becomes the focus of this resource problem (Spier and McFarland, 1996(a)). Paralleling from a natural science perspective ensues the problem of balancing the somatic needs of an animal in its hunt for food/shelter, with perhaps the conflicting activities of exploration, reproduction and play. The requirements for the artificial agent, in this instance, is to coordinate its behaviour to maintain its fuel supply, whilst carrying out other tasks - termed behaviour sequencing, where the solution to the problem is inspired from the adaptive behaviour of animals as discussed in **chapter 1** (Wilson, 1991).

The model presented below discusses the minimum decision-making scenario where there exists a trade-off between refuelling activities and work to ensure the robot is both useful and self-sufficient. The minimum decision-making scenario incorporating trade-off is the two-resource problem (Spier and McFarland, 1997). With this problem, the control of behaviour to achieve a minimum of two conflicting behaviours (for example fuel and goal searching) is studied here. (The one resource problem, although popularly studied by McFarland and co-workers, is not adequate for this work, as it does not provide a trade-off in decision-making).

7.3 The Basic Cycle

There are two basic resources which must be accessible within the environment in order for an agent to be self sufficient: energy R, which must be obtainable and expendable, and is taken here in the case of the robot to be the fuel needed for it to function; and work, W, which may be considered the work of the agent, data collection, goal searching or in this case, the task set by the station crew.

The agent resides in the three dimensional environment developed in previous chapters with, however, these two internal state variables (R and W). The robot's sensors can detect each variable, and must collect each to be self-sufficient. When the robot is refuelling, its energy is increased. When it reaches its goal point, work is done, and so this state variable is increased. At all other times, its state variables decrease. If either level of state variable is permitted to fall to zero, the robot has failed and has effectively 'died'. Hence the robot must collect each to sustain itself. In view of an animal, these state variables may be food and water, however, for this case, the state variables are fuel and the required task (McFarland & Spier, 1997).

It is clear that if the robot were to spend too much time collecting only one state variable, such as carrying out a task, then its internal state would quickly become unstable, for example, the fuel level would drop to a dangerous level. And likewise, if the robot were to break away from its task in order to find fuel too early, it would not be performing optimally. The aim, then, is to maintain a stable homeostasis of the robot's internal state.

In this model, the energy is made available through fuel, collected from a randomly placed depot. The work or task set for the robot is specified as an object collection task where the objects are also randomly placed. A certain level of work must be maintained for the robot to effectively perform. Depleting energy levels, however, necessitates that the robot must break away from its task to search for a refuel depot at some point, conflicting with its work interests. From such conflicts arise a basic cycle of work, consisting of finding work – working - finding fuel – refuelling.

Figure 7.1 represents the basic work cycle used here on the R-W plane. The robot has been designed such that energy is depleted at all times when the robot is not refuelling, i.e. it utilises energy to find work, do work and to find fuel. Likewise, the robot's utility (W) is depleted during all times it is not working, i.e. when searching for fuel, searching for work and refuelling. A careful balance must be achieved to maintain homeostasis.

If an imbalance occurs, such as in **figure 7.2** where the robot takes too long to find fuel, the robot will eventually 'die' as energy levels become depleted. Work levels also become depleted as these are used up when not working. If, as in **figure 7.3**, the robot takes a longer than usual time to find work (i.e. collect an object), the robot will also eventually 'die' as work levels become depleted. Energy levels also become depleted as these are used up when not refuelling.

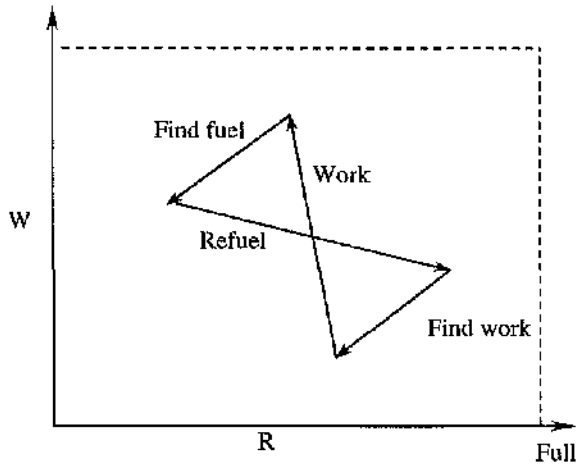


Figure 7.1: An example of the basic work cycle of: work - find fuel – refuel, placed on the R-W plane.

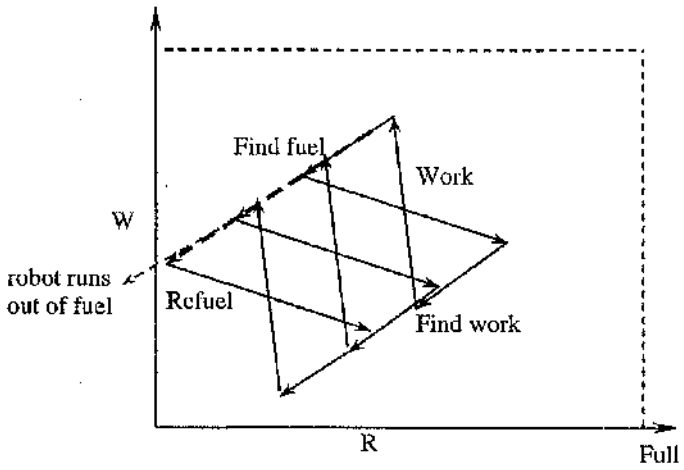


Figure 7.2: An example of an unstable work cycle where the robot runs out of fuel.

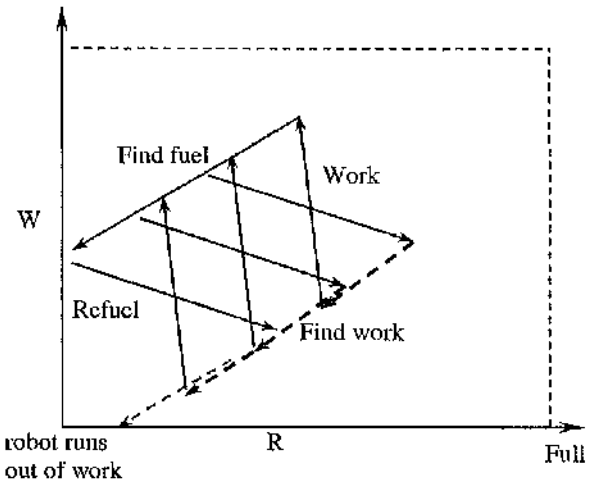


Figure 7.2: An example of an unstable work cycle where the robot runs out of fuel.

7.4 The State Space Model

Sibly and McFarland originally proposed the concept of a state space model for agents in 1974, which was further developed in 1981 by McFarland and Houston. This concept presents the definition of the agent through a minimal set of internal variables, which describes its state completely (Sibly and McFarland, 1974) (McFarland and Houston, 1981). The physiological state variables that would identify

a biological system could possibly be defined as hunger, thirst, temperature etc. However, for the case of the robot, we can identify energy, internal temperature or tasks.

Figure 7.4 provides an example of a two-dimensional state space for a biological system. The variables, here chosen as oestrogen levels and temperature, rest within a Euclidean vector space as its orthogonal axes. The current physiological state, represented as P, rests within the boundary surfaces, which define the limits to the state space. The boundaries mark the edge of prohibited areas where the agent is either unable to reach (such as negative hormone levels), or which in doing so, would result in its 'death' (such as extreme temperature) – these fatal boundaries are known as the 'lethal limits'.

Position P, (or vector $\underline{p}$) represents the current physiological state, whereas T depicts a possible trajectory the agent could take. It is the aim of the agent to maintain its physiological homeostasis despite internal perturbations (changing hormone levels alters internal temperature; eating increases thirst levels) or indeed environmental perturbations (the presence of a mate; a drop in outside temperature).

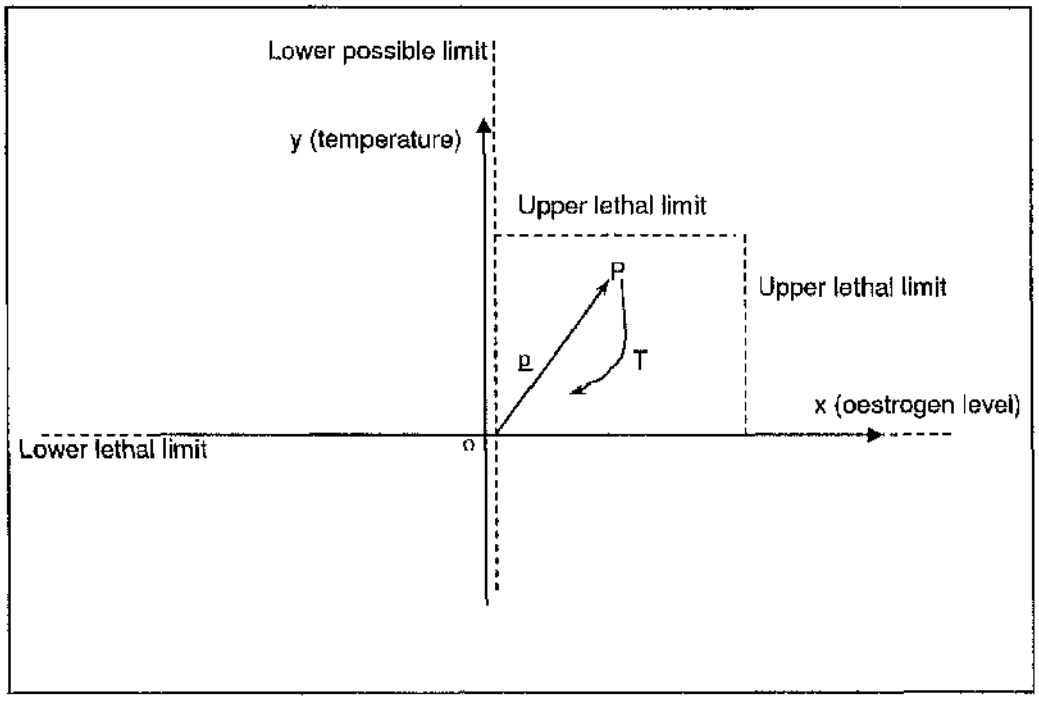


Figure 7.4: A biological two-dimensional state space with local origin o.

A robotic equivalent can be seen in **figure 7.5**, whereby the axes are assigned to fuel levels and data collection. Data collection is given as the task desired of the robot, whilst maintaining fuel levels is the necessary function required to sustain itself. In this case the lower boundary for the fuel variable becomes a lethal limit (since having no fuel (y=0) results in the 'death' of the robot) with the upper boundary governed by the capacity of the fuel tank. Any behaviour the robot may exhibit can be described by its trajectory taken within the state space.

The robot must function to maintain homeostasis of these state variables by regularly checking its state co-ordinates and manipulating its behaviour accordingly to maintain a desired position and hence maintain local equilibrium. It can be seen from **figure 7.2** that the unstable work cycle drives the robot to its lethal limits, whereas in **figure 7.3**, spending a shorter time to find fuel drives the robot to its upper possible limits.

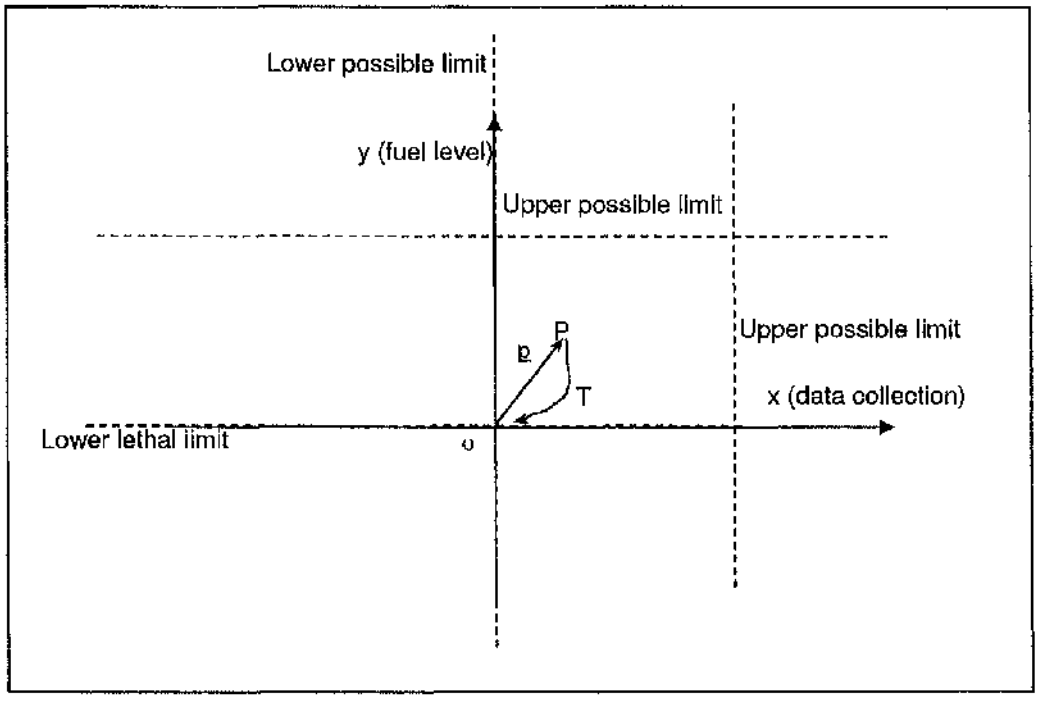


Figure 7.5: A robotic two-dimensional state space with local origin o.

7.5 The Cue-Deficit model

Sibly and McFarland (1975) developed a decision-making process to maintain homeostasis within the lethal limits of the state variables. This process is influenced by both the *deficits* of the state variables, and stimuli from the environment. The stimuli act as a *cue* to available resources that would affect the position of the state variables, hence the *Cue-Deficit model*.

For the state space example above, the decision to perform a behaviour is made by calculating the motivation to perform all possible behaviours, in this case data collection or refuelling, and choosing the behaviour with the highest motivation.

The motivation for each behaviour is calculated by multiplying the state variable deficit with the resource cue to provide a strength value for the motivation. A behaviour is chosen by comparing the strengths of all the behaviours and choosing the one which exhibits the highest motivational strength.

Figure 7.6 demonstrates a series of motivational isoclines, whereby the isoclines connect equal motivational strengths by connecting all the possible combinations of cue-deficit products. Any one point, sitting on a motivational isocline has an equal motivational tendency as any other point. The cue deficit model chooses its behaviour on the basis of the one with the greatest motivational tendency.

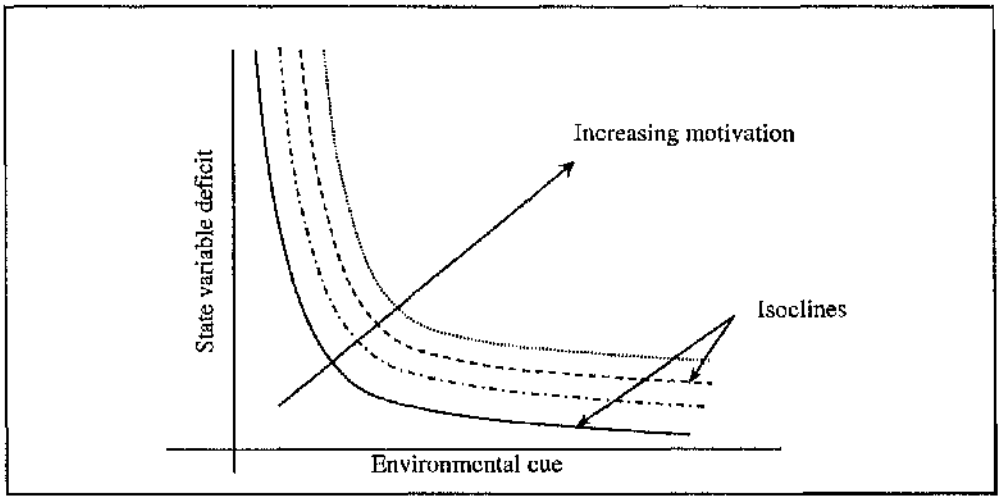


Figure 7.6: Motivational Isoclines

The motivation based cue-deficit model draws similarities from the popular quadratic cost function model also devised by Sibly and McFarland (1976) to obtain optimal behaviour. The planning method of the cost function model connects the availability (or density), r, of the resource to the accessibility (the ease in obtaining the resource), k, to the deficit (hunger, h, or thirst, t, etc for a biological system) and may be represented for a two resource problem as:

if $h.r_h.k_h > t.r_t.k_t$ then eat,
if $h.r_h.k_h < t.r_t.k_t$ then drink,

where the subscript h denotes hunger, and likewise t denotes thirst (Spier & McFarland, 1997(b)).

Similarities can be drawn between the reactive cue-deficit model and the planning methods of the cost function model if it may be assumed that the cue for a particular environmental resource that an agent receives may influence the rate of gain of that associated state variable. The cue may be associated with the $r_h.k_h$ term of the quadratic cost function, thus validating the multiplication of the cue-deficit terms.

7.6 The Algorithm

The cue-deficit model is considered and compared for each of the resources in the current model, where the tasks are chosen to be refuelling and data collection (reaching a goal). Reaching a goal quickly is however, vital to the sustained usefulness/life of the robot and when reached, the goal remerges in a new random spot, renewing the task. Likewise, when the robot has refuelled, the refuel depot relocates elsewhere. A similar example of this task is discussed by Birk (1998) where he makes reference to reaching 'competitors' or small lamps connected to the same global energy source as the charging station. If the robot knocks against the competitors the lamps dim and allow additional energy to be available at the station, and hence reaching these objects is vital for self-sustenance and forms the working task of the robot.

Since reaching a goal (data collection point) is chosen to be the essential function of the robot in this case, failure to do so, within a certain time period, signifies the robot has failed its task. However, failure to refuel before the onboard fuel has depleted means the robot has failed to self-sustain. Given these conflicting interests, the cue-deficit technique is used to decide its behaviour.

The strategy for the cue-deficit technique, applied in this example, requires calculating the motivational strength of each of the state variables and subsequently choosing that behaviour for which the strength is greatest, such that:

if

$\text{Deficit}_A * \text{Cue}_A > \text{Deficit}_B * \text{Cue}_B$

then

choose A

else

choose B

where A and B represent the resources.

The deficit for the fuel resource is calculated to be the total fuel used since the last refuelling point, the calculation for which is shown below. The deficit for the working behaviour (e.g. goal searching/ data collection) is regarded as a depletion of utility, in this case regarded as the time passed since the last goal was accomplished, and is calculated proportionate to the time expended since the previous goal was attained. The cue for both behaviours is inversely proportional to the distance the robot is from each resource, i.e. the distance the robot is from the refuelling station or other goal, such that:

$$\text{Motivation}_{\text{refuel}} = \text{Deficit}_{\text{refuel}} * \text{Cue}_{\text{refuel}}$$

$$= b\frac{\text{fuel\_used}}{\text{distance\_to\_goal}} \qquad (7.1)$$

$\text{Motivation}_{goal} = \text{Deficit}_{goal} * \text{Cue}_{goal}$

$$= b\frac{\text{depletion\_of\_utility}}{\text{distance\_to\_fuel}} \qquad (7.2)$$

where b is a scaling constant, taken here for ease of calculation to have the same value in both **equation 7.1** and **equation 7.2.**

The robot is assumed to have a fixed, initially full tank of fuel. At each time-step, the propellant mass needed to carry out the desired manoeuvre is calculated and subtracted from the fuel mass remaining. A warning is indicated if the fuel levels drop to a low level, chosen to be one quarter of a full tank. The robot, in later chapters, is given a human control capability, incorporated in the form of joystick control of the robot's three degrees of freedom, allowing human influence on the robot at any time, if desired. The warning indicator of low fuel levels thus provides the operator time to intervene and guide the robot to the fuel depot using manual control.. The amount of propellant used since the previous refuel is recorded after each time-step and is used as the deficit for the refuel motivation.

The fuel consumption is calculated from the following:

The change in robot mass due to consumption of element of fuel, Δm, can be related to the change in speed, Δv, as illustrated in **figure 7.7**:

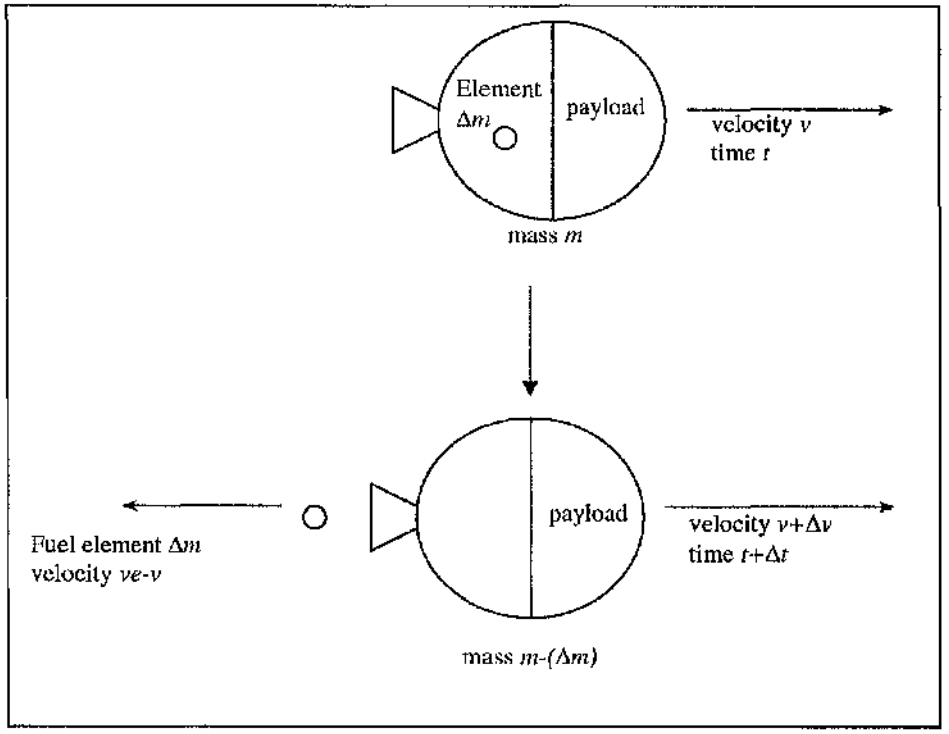


Figure 7.7: The change in robot mass due to consumption of element of fuel, Δm

In time, Δt, mass element, Δm, is lost with exhaust speed v_e. The vehicle gains forward speed Δv.

From the conservation of linear momentum it is clear that:

$$mv = \big(m-(-\Delta m)\big)(v+\Delta v)-(-\Delta m)(v_e-v) \tag{7.3}$$

So that

$$m\Delta v = -v_e\Delta m \tag{7.4}$$

Defining the propulsion specific impulse I_{sp} as the amount of momentum gained per unit weight of propellant used:

$$I_{sp} = \frac{(\Delta m)v_e}{g(\Delta m)} \tag{7.5}$$

It can be seen that

$$I_{sp} = \frac{v_e}{g} \tag{7.6}$$

Then substituting **(7.6)** into **(7.4)** and rearranging gives:

$$\Delta m = \frac{m\Delta v}{I_{sp}g} \tag{7.7}$$

where Δm is the propellant mass used, Δv is the change in velocity, g is the gravitational constant, m is the robot mass and I_{sp} is the propellant specific impulse (thrust/propellant weight flow rate).

As detailed in **chapter 2** the above data is given as:

$m = 2.5\text{kg}$ $\quad g = 9.81\text{ms}^{-2}$ $\quad I_{sp} = 50\text{s}$

Hence **equation (7.7)** becomes:

$$\Delta m = \frac{2.5(\Delta v)}{9.81(50)} \tag{7.8}$$

i.e

$$\Delta m \cong 0.0051(\Delta v) \tag{7.9}$$

The distance of the robot to the refuel depot is calculated by assuming the depot emits a signal, such as an ultrasound signal detailed in **chapter 2**, which can be detected by the robot's proximity sensors. The strength of this emitted signal is, for

simplicity, assumed to be calibrated inversely proportional to the distance from the goal centre such that there is an inverse gradient of maximum value at the source and reducing inversely outward.

The distance from the refuel depot may then be calculated as:

$$r = \sqrt{\left(x - x_{refuel}\right)^2 + \left(y - y_{refuel}\right)^2 + \left(z - z_{refuel}\right)^2} \qquad (7.10)$$

and the inverse of this then becomes the cue for the refuel state variable within the motivation calculation.

The cue and deficit terms involved in finding the goal are calculated in exactly the same manner. The deficit for finding the goal is calculated as the fuel deficit, resetting to zero each time the goal is found. In real terms, the desire to find the goal increases at the same rate as the desire to refuel, which in turn, is equal to the rate of consumption of fuel. Ensuring the rates are initially equal eases analysis of the results, and from here the ratio of rates may be simply altered by changing the ratio of the scaling constant k, initially set to 1.

The cue for finding the goal is calculated, for simplicity, above as:

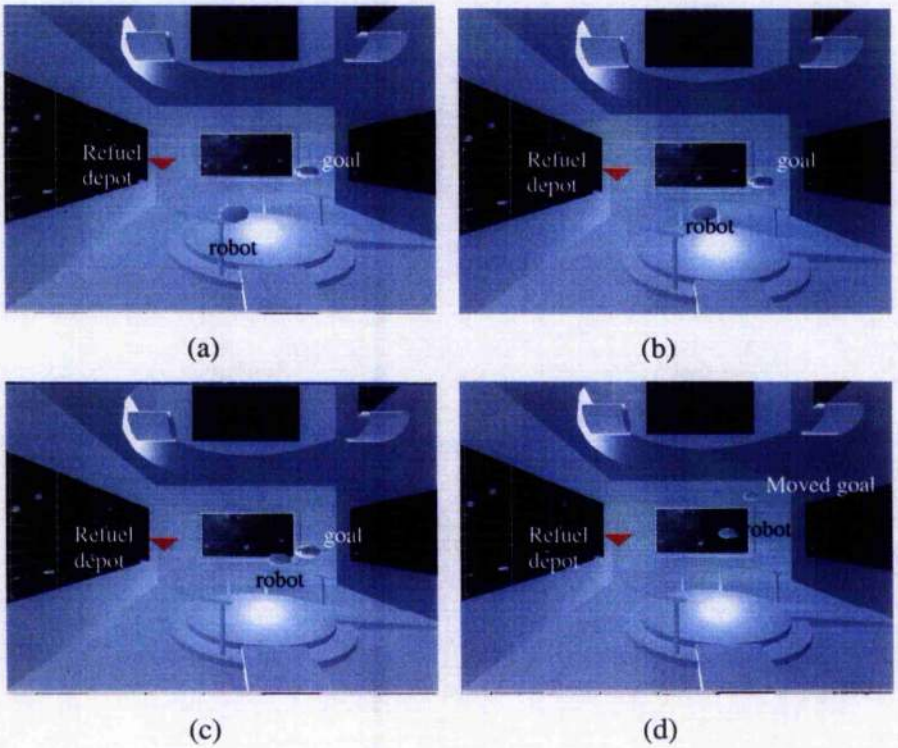
$$Cue_{goal} = \frac{1}{r} \qquad (7.11a)$$

$$\text{where } r = \sqrt{\left(x_{start} - x_{goal}\right)^2 + \left(y_{start} - y_{goal}\right)^2 + \left(z_{start} - z_{goal}\right)^2} \qquad (7.11b)$$

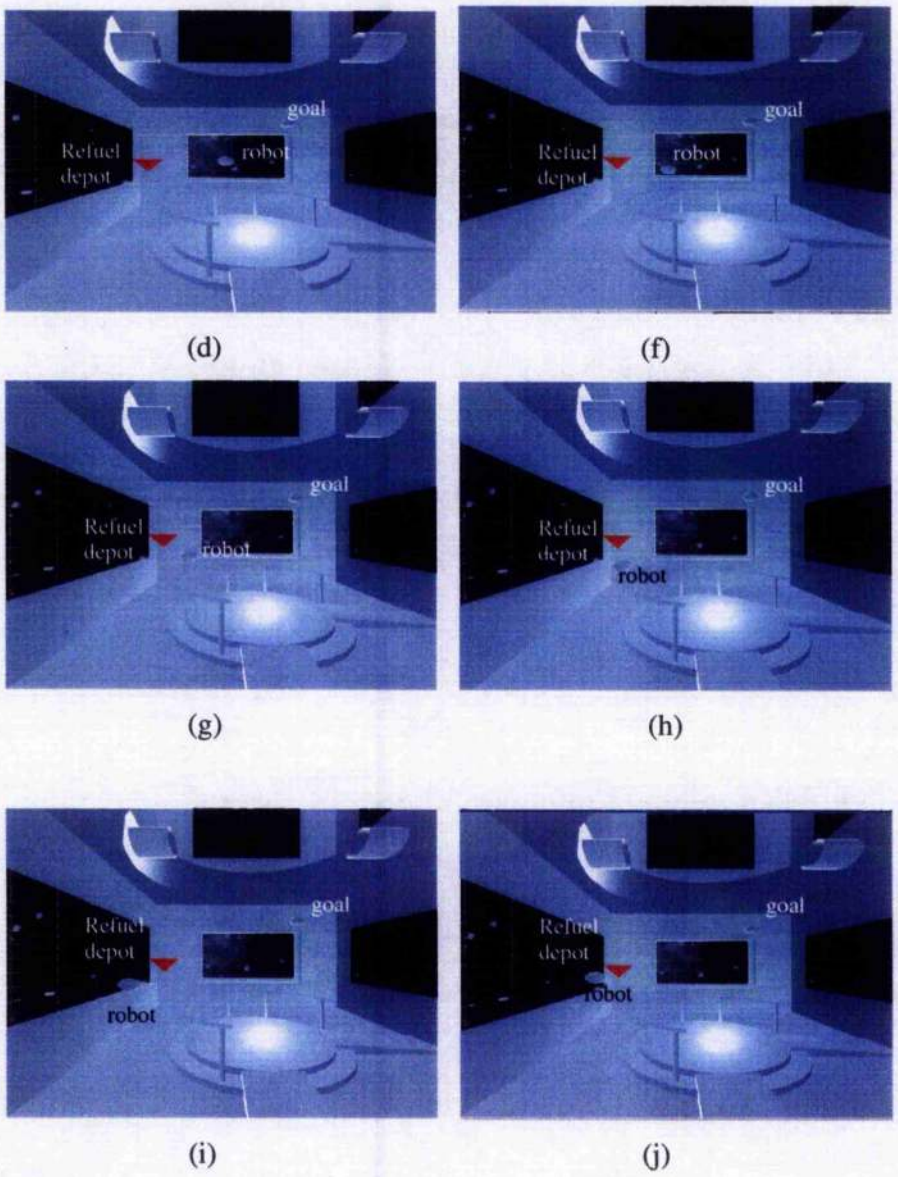
7.7 Testing the Cue-Deficit Model

The spacecraft environment constructed in previous chapters provides the basis for testing the model. The robot is immersed in the environment and, at each time-step, calculates the motivation required to carry out each of the tasks: refuelling and finding the goal. The robot proceeds with the task for which exists the greatest motivation, whilst avoiding possible collision hazards and controlling maximum speed.

Figure 7.8 shows a series of screen shots depicting the robot's behaviour given the above tasks. Initially, the robot proceeds with its task, in this instance reaching the goal, represented as a small sphere in the right of the first three figures. Upon reaching the goal, the goal relocates (found in the top, far right of figure (d)) and the robot now attempts to find the refuel depot, which is closer (represented by the triangle to the left of figure (d) and which it reaches in figure (i)).



Figures 7.8(a)-(j): Snapshots displaying the robot's motivation based behaviour.



Figures 7.8(a)-(j): Snapshots displaying the robot's motivation based behaviour.

In **figure 7.9**, the fuel reserves and utility are plotted against time. There is a gradual decline in fuel as the robot proceeds with its tasks. After a short period of time it comes in close contact to a fuel depot, which, although setting off with a full tank of fuel, uses the opportunity of being close to the fuel depot to refuel. This opportunistic behaviour is characteristic of the cue-deficit method, but is extremely difficult to endow in classical A.I. systems. When the refuel station is *reached*, there is an expected jump in fuel reserves symbolic of filling the tank. A similar plot is

made for utility. Again, as the robot *achieves* its task (reaches the goal, or collects an object) its utility is increased, whereas at all other times, whether engaged in the task or looking for the refuelling station, the utility is slowly drained.

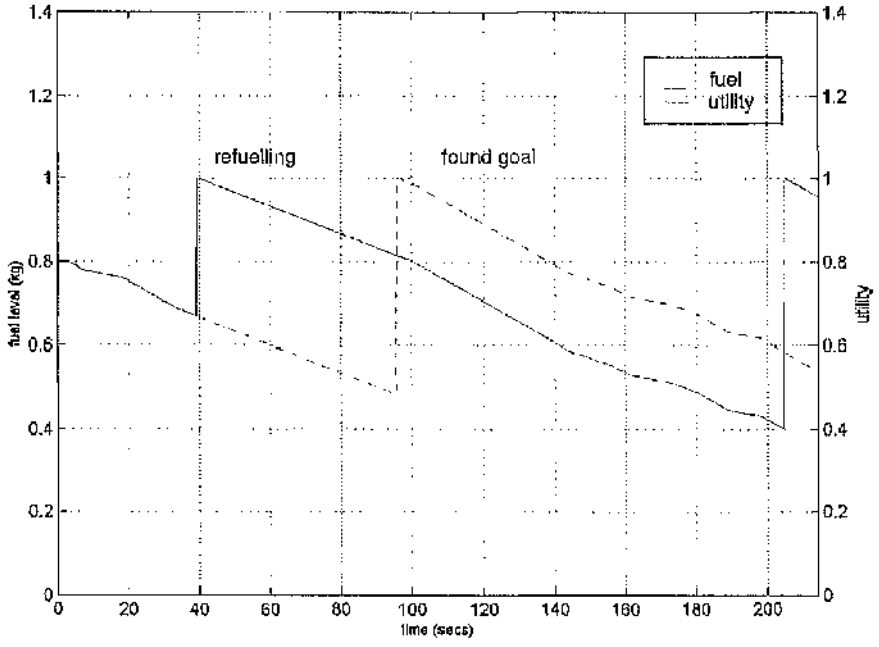


Figure 7.9: Plot showing fuel level and utility over time.

In **figure 7.10**, the *motivation* to perform each of the two behaviours is plotted against time. In each case there is a gradual increase in motivation with time due to consumption of the resource, combined with moving closer to a new source. This increase in both cue and deficit is manifest as increased motivation. The sharp drops in motivation occur when the robot has found a resource, which, when consumed, then reappears elsewhere. The cue is, therefore, reduced by relocation, and the deficit is reset to zero. Any small dips in motivation can be explained as movement of the robot temporarily away from the source it is seeking, perhaps due to obstacle avoidance strategies, or overrun of its guidance strategies.

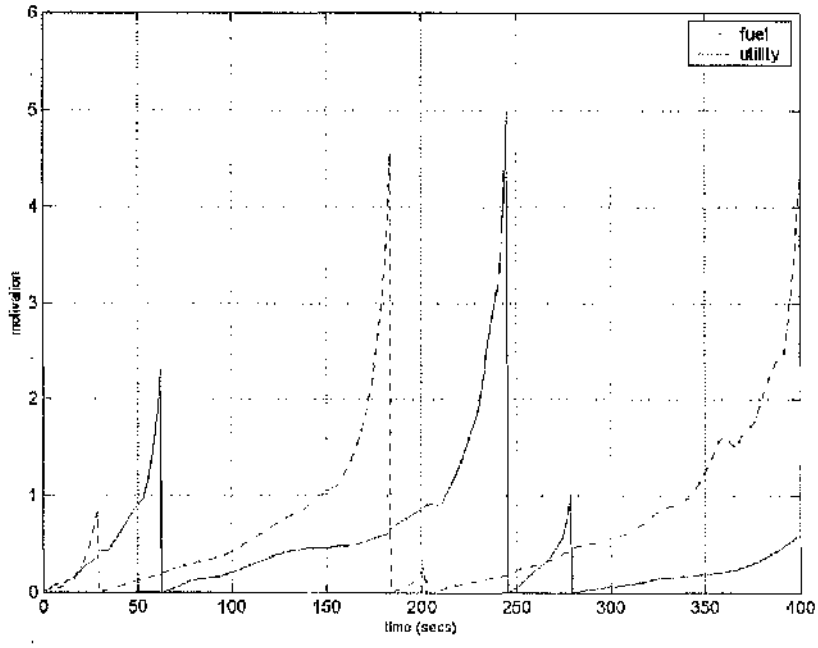


Figure 7.10: Plot showing motivation against time for each of the two resources.

7.8 Conclusion

This chapter discussed the problem faced by artificial agents, to maintain self-sufficiency while performing the design tasks required of it. The model presented discussed the minimum decision-making scenario where there exists a trade-off between self-sustaining activities and work to ensure the robot is both useful and self-sufficient. This minimum decision-making scenario, *the two-resource problem*, controlled the agent's behaviour to achieve a minimum of two conflicting resources, here chosen to be refuelling and finding the goal. From this arose the concept of the basic cycle of work, consisting of work - finding fuel – refuelling. The robot was designed such that energy is depleted when the robot is not refuelling, (both do work and to find fuel) and the robot's utility is depleted during all times it is not working.

In attempt to assure self-sufficiency, the *cue-deficit* model was approached where the decision to perform a behaviour was made by calculating the motivation to perform all possible behaviours, in this case, reaching the goal (e.g. an astronaut) or

refuelling, and choosing the behaviour with the highest motivation. The motivation for each behaviour was calculated by multiplying the state variable deficit with the resource cue to provide a strength value for the motivation. The behaviour was chosen by comparing the strengths of all the behaviours and choosing the one with the highest motivation strength.

Chapter Eight: Integration

Detective Del Spooner: [to Sonny] "It's a human thing; you wouldn't understand."

I, Robot

8.1 Introduction

The robot model has, so far, integrated a number of algorithms within a developing control methodology, equipping the robot with an ability to carry out a series of chosen mission tasks, and tested these algorithms by immersing the robot in a three-dimensional virtual environment, representing a potential space station module.

In addition, the problem of balancing the desired tasks whilst maintaining functionality was addressed in the previous chapter. Sequencing the robot's various activities to provide efficiency, self-sufficiency and functionality are essential for a useful working robot, made possible by incorporating cue-deficit techniques derived from animal behaviour studies. The robot model is now near to completion. However, it is evident that, so far, each behaviour is given a state of either *on* or *off*, such that the transition between one behaviour and the next is instantaneous. This is manifest in sometimes rather clumsy motion. In this chapter, a solution is developed to provide a smooth transition between behaviours.

This stage involves the application of a weighting function for each behaviour. This function, unlike previous weighting constants, is chosen to be variable, with a magnitude that is influenced by vehicle state. This allows a seamless transition

between one behaviour and the next as appropriately governed by the sensed environment.

In addition, a human controller is interfaced with the design, providing the now completed control methodology with human-robot synergy, and so improving versatility. This human control is manifest in the form of a force-feedback joystick, enabling an operator to change or influence the overall behaviour of the robot from fully autonomous control. Finally, the system is faced with a number of scenarios aimed at testing the robot's abilities and efficiency in various situations.

8.2 The Weighting Function

In previous sections, the weighting, λ, for each behaviour was applied as a constant, and chosen, in effect, to scale the behaviour, or to provide a ratio between one algorithm and the next. This, however, has the drawback of creating an abrupt transition from one behaviour to another as the state of the vehicle changes, causing a rather blunt trajectory. A more versatile, smooth transition requires a weighting function that will vary in accordance with state changes.

For the purpose here, the weighting function is chosen to be a function of the ratio of the distance to the nearest sensed obstacle (d_{near}), to the sensor's obstacle detection range. Hence enabling the weighting to vary with sensed state.

Figure 8.1 shows the model design, illustrating the constant weighting function currently used. In **figure 8.2** the desired variable weighting function is shown, which will be developed in this chapter.

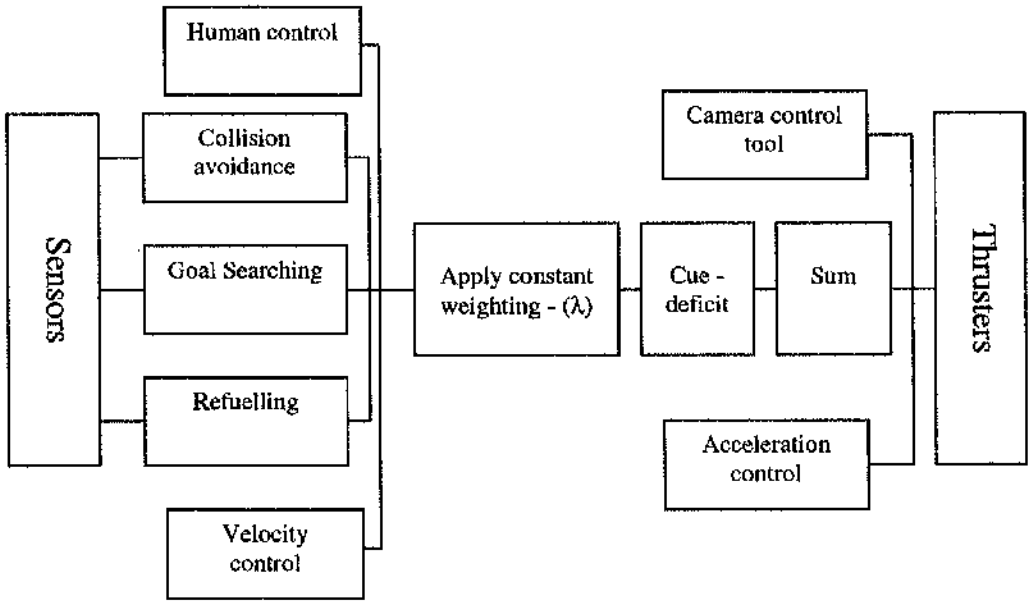


Figure 8.1: Control model with constant weighting.

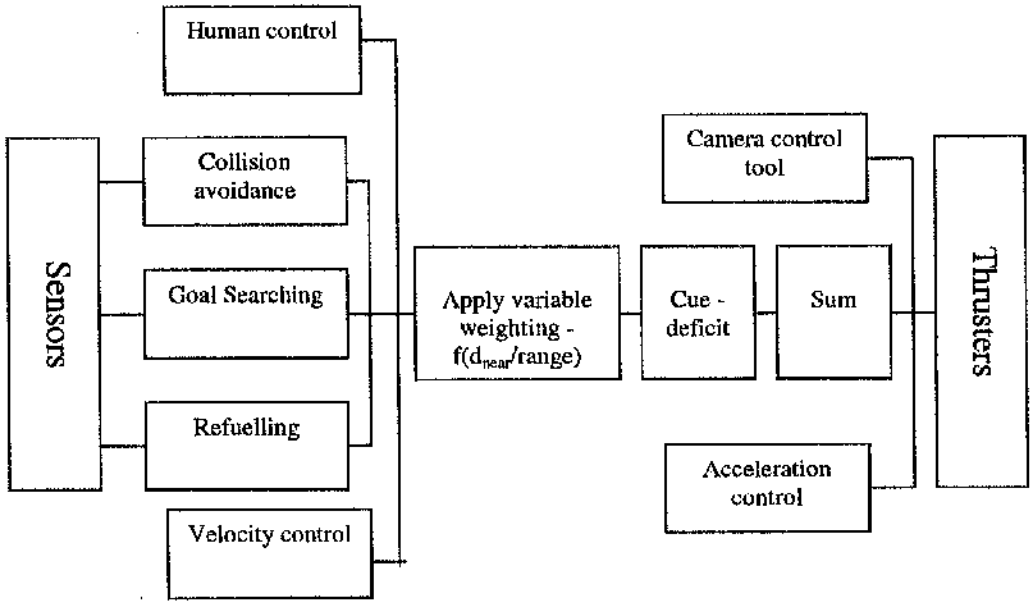


Figure 8.2: Control model with variable weighting.

8.2.1 *Goal-searching*

For the case of the goal-searching algorithm, the weighting is directly proportional to the ratio of the distance to the nearest sensed obstacle, to the sensor's obstacle detection range.

Such that:

$$\lambda_{goal} = \frac{d_{near}}{range} \tag{8.1}$$

and from this, the final goal-searching acceleration becomes:

$$\underline{a}_{goal\_final} = \lambda_{goal} \cdot \underline{a}_{goal} \tag{8.2}$$

where:

λ_{goal} = goal acceleration weighting function,

$range$ = obstacle detection sensor range,

$\underline{a}_{goal}$ = goal-searching acceleration before weighting,

$\underline{a}_{goal\_final}$ = final goal-searching acceleration after the application of the weighting function.

This ratio is chosen such that as the distance to a detected obstacle is reduced, i.e. as the robot approaches a possible collision hazard, the weighting governing the goal acceleration is gradually reduced. This temporarily minimises the influence of this behaviour $\underline{a}_{goal\_final}$ on the chosen path of the robot, making way for, for example, collision avoidance strategies to dominate the robot's behaviour.

When the nearest sensed obstacle is detected just on the sensor's horizon, then the ratio becomes 1, and hence the goal-searching weighting is at its maximum, enabling a maximum goal-searching behaviour, since it is decided the obstacle no longer poses a collision threat when on the sensor range horizon. The ratio would never exceed one, since the robot would never detect an obstacle outside its sensor range.

Figure 8.3 was produced to demonstrate the variation in weighting of the goal-searching behaviour (on the x-weighting) with distance detected to the nearest obstacle, over a period of 450 seconds. The robot, equipped with a sensor object detection range of 3 metres, was released in an environment containing potential collision hazards. From **figure 8.3,** it is noted that the goal searching weighting mirrors the distance detected to the nearest obstacle. If no obstacle can be detected, the sensors record a maximum value of 3 metres, the sensor horizon, as a default, and the goal-searching behaviour is given a maximum weighting value of 1, fully operational. However, when the robot approaches an obstacle within the sensor horizon, the goal-searching weighting gradually decreases, de-rating this behaviour to make way for collision avoidance strategies. Likewise, as the robot retreats from the collision hazard, the goal-searching weighting gradually increases, allowing the robot to resume its search for the goal point, having safely passed the collision hazard.

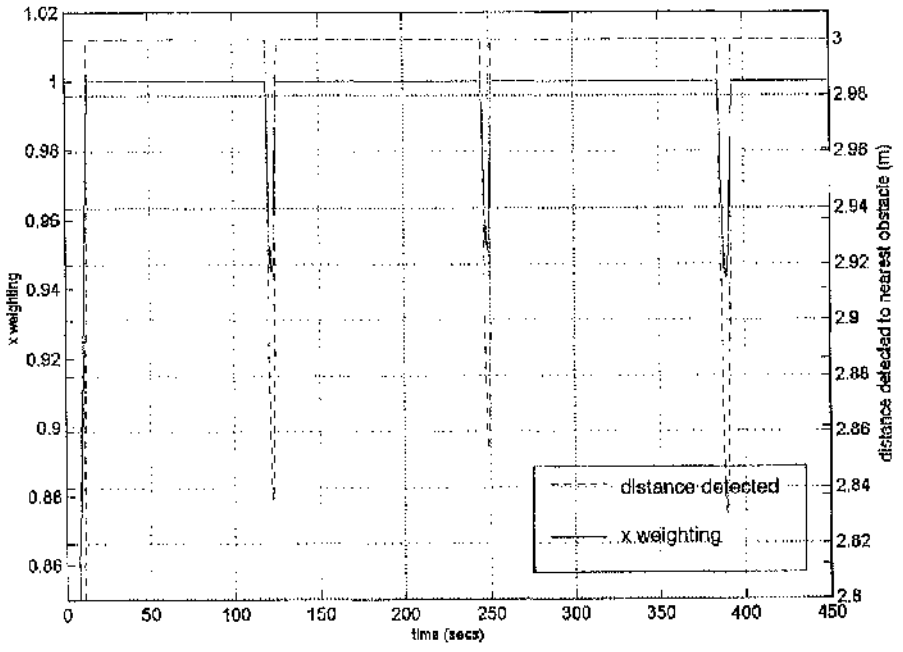


Figure 8.3: Variation in goal-searching weighting with distance to the nearest detected obstacle.

8.2.2 *Obstacle_avoidance*

The collision avoidance weighting has been chosen as:

$$\lambda_{avoid} = 1 - \frac{d_{near}}{range} \tag{8.3}$$

and from this, the final collision avoidance acceleration becomes:

$$\underline{a}_{avoid\_final} = \lambda_{avoid} \cdot \underline{a}_{avoid} \tag{8.4}$$

where

$\underline{a}_{avoid}$ = obstacle avoidance acceleration before weighting

λ_{avoid} = avoidance acceleration weighting function

$\underline{a}_{avoid\_final}$ = final avoidance acceleration after the application of the weighting function.

This chosen weighting has an opposite effect from the goal-searching weighting, such that, as a collision hazard is approached, $\frac{d_{near}}{range}$ gradually decreases, and subsequently the overall collision-avoidance weighting increases, (hence the collision avoidance behaviour is increased). This would continue effectively until impact, where d_{near} would become zero, the ratio would then become unity and the collision-avoidance weighting maximised, however the result of having an increasing collision avoidance mechanism would prevent impact from occurring.

At such a point when the nearest sensed obstacle is detected just on the sensor's horizon, the ratio $\frac{d_{near}}{range}$ becomes one, and therefore the weighting, $1 - \frac{d_{near}}{range}$, becomes zero, diminishing the collision avoidance acceleration, since the collision hazard has passed.

Figure 8.4 plots the weighting of the obstacle avoidance behaviour against distance to a collision hazard, on the x-weighting axis, over a period of 450 seconds. Again the vehicle is equipped with a sensor object detection range of 3 metres, and released in an environment containing a number of potential collision hazards. It is shown that the obstacle avoidance weighting, traced in blue, is zero whenever it is the case that no hazard can be detected. i.e. obstacles are out-with the vehicle's sensor range, and so decided to be of no danger (sensors record a value of 3 metres when no obstacle is in view). However, as the robot approaches an obstacle, sensed *within* the sensor horizon of three metres, the collision avoidance weighting begins to increase, causing the robot to retreat from the obstacle, avoiding a potential collision. Likewise, as the robot successfully retreats from the collision hazard, evident on the plot by an increase in the detected distance to a nearby obstacle, there is a simultaneous decrease in the collision avoidance weighting until it again becomes zero, - when the obstacle has been passed.

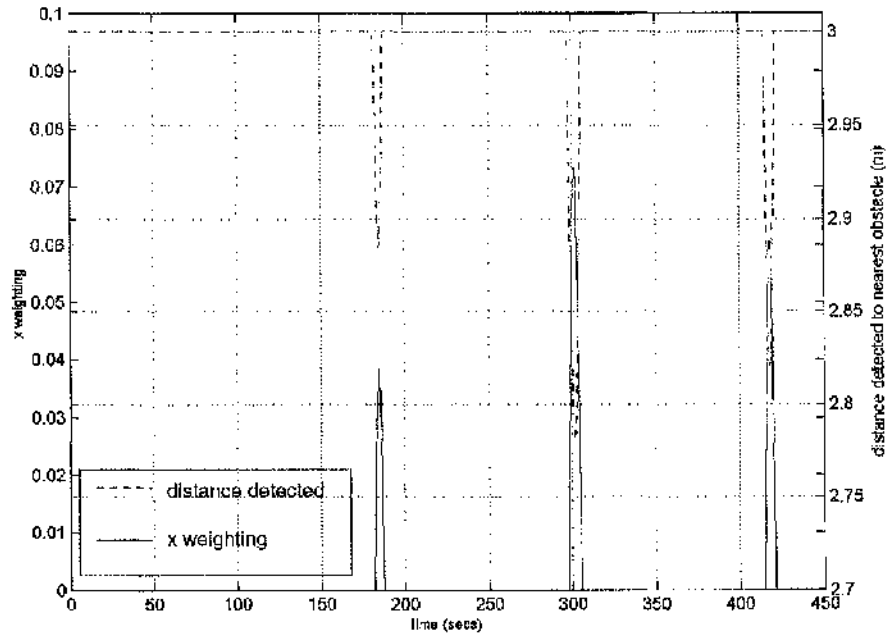


Figure 8.4: Variation in obstacle avoidance weighting with distance to the nearest detected obstacle.

8.2.3 *Refuelling*

For the case of the refuelling algorithm, the weighting is, as with the goal searching algorithm, directly proportional to the ratio of the distance to the nearest collision hazard and the sensor's obstacle detection range.

Such that:

$$\lambda_{refuel} = \frac{d_{near}}{range} \tag{8.5}$$

where λ_{refuel} = the refuel weighting.

And from this, the final refuel acceleration becomes:

$$\underline{a}_{refuel\_final} = \lambda_{refuel} \cdot \underline{a}_{refuel} \tag{8.6}$$

where

$\underline{a}_{refuel}$ = refuel acceleration before weighting

$\underline{a}_{refuel\_final}$ = final refuel acceleration after the application of the weighting function.

Again, this ratio is chosen such that as the robot approaches a possible collision hazard, the weighting governing the refuel acceleration is gradually reduced. This temporarily minimises the influence of this behaviour on the chosen path of the robot, by reducing the final refuel acceleration, to allow collision avoidance strategies to dominate the robot's autonomy until the collision hazard has passed.

When the collision hazard can be *only just* detected on the sensor's horizon, then the ratio becomes 1, (since $d_{near} = range$) and hence the refuel weighting is at its maximum. This manifests maximum refuelling behaviour, since it is decided that the obstacle no longer posses a collision threat when on the sensor range horizon. The ratio would never exceed one, since the robot would never detect an obstacle outside its sensor range.

Figure 8.5 again demonstrates the variation in weighting of the refuelling behaviour (on the x-axis) with distance detected to the nearest obstacle, over a period of 450 seconds. The vehicle, as before, was equipped with a sensor object detection range of 3 metres. It is shown, as with the goal-searching behaviour, that the refuelling weighting is maximised when all collision hazards lie out-with the vehicle sensor range of 3 metres, i.e. the refuelling motivation is fully operational. However, when the robot approaches an obstacle, the refuelling weighting gradually decreases, reducing the motivation to find a refuel station whilst there is an obstacle in its pathway. Likewise, as the robot retreats from the collision hazard, the refuelling weighting gradually increases to a maximum value when the object lies outside the sensor horizon, allowing the robot to resume its hunt for fuel since the collision hazard has passed.

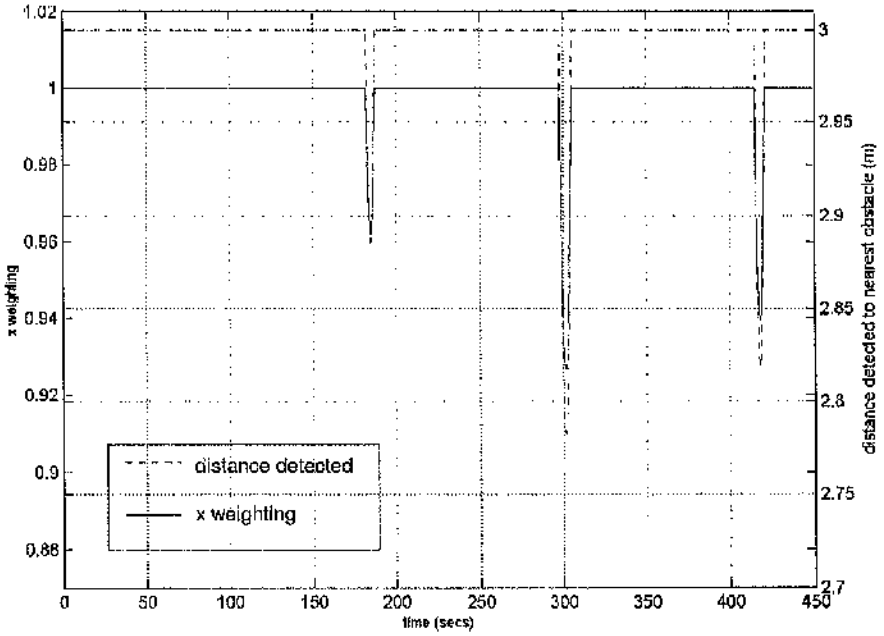


Figure 8.5: Variation in refuelling weighting with distance to the nearest detected obstacle.

8.3 Combining the Weighting Functions

Having developed and tested the weightings for each behaviour, the total acceleration for the robot is chosen to be the combination of each of the individual behavioural accelerations. Such that:

$$\underline{a}_{total} = \underline{a}_{goal\_final} + \underline{a}_{avoid\_final} + \underline{a}_{refuel\_final} \tag{8.7}$$

$$\text{i.e. } \underline{a}_{total} = \lambda_{goal} \cdot \underline{a}_{goal} + \lambda_{avoid} \cdot \underline{a}_{avoid} + \lambda_{refuel} \cdot \underline{a}_{refuel} \tag{8.8}$$

$$\text{i.e. } \underline{a}_{total} = \frac{d_{near}}{range} \cdot \underline{a}_{goal} + \left(1 - \frac{d_{near}}{range}\right) \cdot \underline{a}_{avoid} + \frac{d_{near}}{range} \cdot \underline{a}_{refuel} \tag{8.9}$$

Combining the behaviours in this way creates a smooth, seamless transition between one behaviour and another. For example, on approach to an obstacle, there is a steady increase in the weighting on the collision avoidance behaviour, combined with a simultaneous decrease in the refuelling and goal searching weightings, until the collision hazard has passed, being out of sight of the sensor horizon, at which point the other behaviours fully resume.

Figure 8.6 superimposes the variation of each behaviour weighting over time during a simulation run where all three behaviours were combined to produce the robot's overall reaction to the environment.

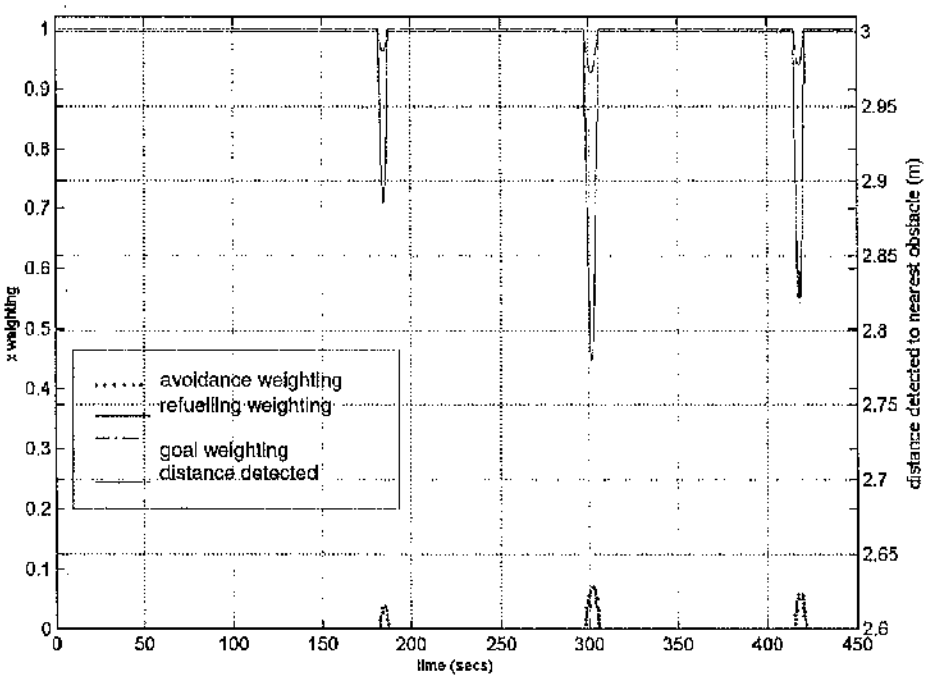


Figure 8.6: Variation in all the weightings with distance to the nearest detected obstacle.

It is evident from the graph that the simultaneous and gradual change in weightings as the robot detects changes in the sensed environment, such as the detection of an obstruction to its path. This graph demonstrates the smoothness of the transition between one behaviour and the next. Note the increase in the collision avoidance weighting with accompanying drop in goal-searching and refuelling weightings as the distance to an obstacle is reduced, at 180 seconds, 300 seconds and 420 seconds.

8.4 Human Interface

The functioning model is now enhanced to support human control capabilities, by integrating a final behaviour within the developed network in the form of a manually controlled joystick with force-feedback effects.

Incorporating tuneable autonomy enhances the system by allowing an operator to influence control of the robot vehicle at any time, if desired, enabling an integrated human-robot synergy.

By passing part of the manual control of a free-flying robot over to the autonomous control system, in the case of this developed controller (including autonomous control of six degrees of freedom, both rotational and translational) the undertaking of basic mission tasks, and maintaining functionality and efficiency, effectively allows a reduction in the operational complexity of the overall control of the freeflyer, de-skilling manual control and enabling the operator to proceed with the control of more high level. In this instance, the control methodology functions to deskill the operation of the vehicle, allowing the operator manual control of the robot's translational motion, with the weighted autonomy providing collision avoidance etc. Such deskilling will be important for long duration, crewed deep space missions, for example, for external inspection and maintenance.

8.5 Integrating the Human Behaviour

A human control capability is incorporated in the form of joystick control of the robot's three translational degrees of freedom - up/down, left/right, forward/back, taken on the robot's x, y, z body axes.

The joystick mechanism has force-feedback effects such that as greater force is applied to the control column, there is a proportional increase in the level of acceleration desired. This is incorporated into the control system as a new behaviour - the human input behaviour, as shown in **figure 8.7**. This behaviour acts in a similar manner to each of the other behaviours and summed accordingly.

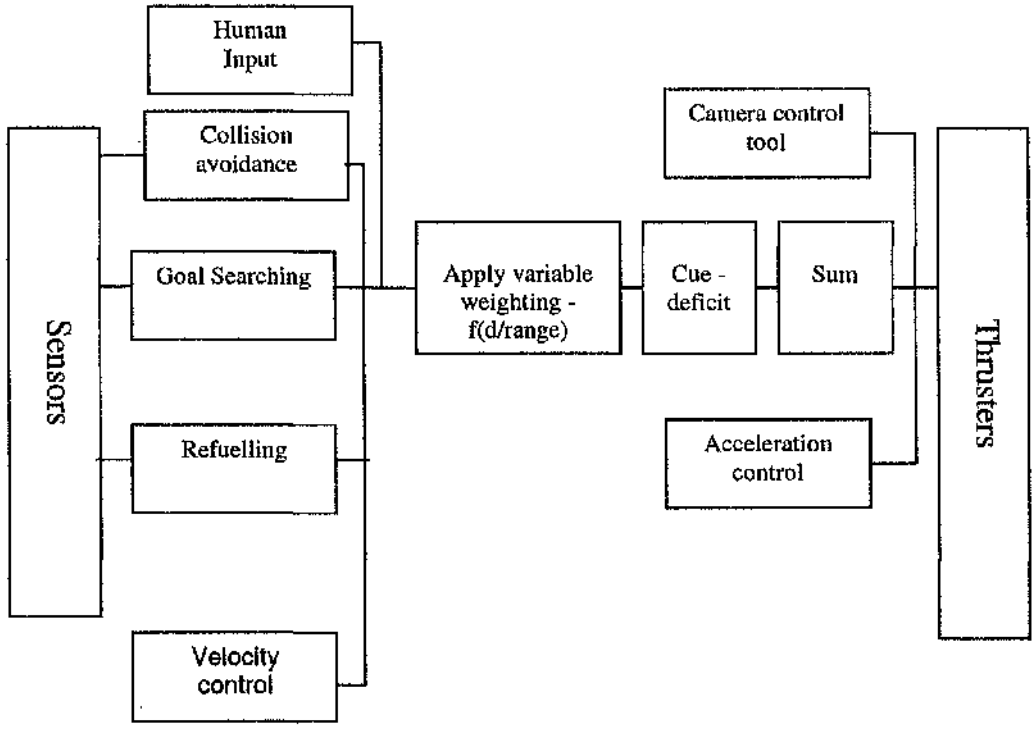


Figure 8.7: Control methodology with variable weighting.

The desired acceleration produced from the human-input control algorithm is defined in this portion of pseudo-code:

```
void joystick(buttons, x, y, z)
{
    while (button1pressed){thrustup = thrustup +1;}
    while (button1 not pressed){thrustup = 0;}
    while (button 2 pressed) {thrustup = thrustup -1;}
    while (button 2 not pressed) {thrustup = 0;}
    a_joystick_x = 0.1(thrustup)

    while (controller pushed to right) {thrustleft = thrustleft -1;}
    while (controller not pushed) {thrustleft = 0;}
    while (controller pushed to left) {thrustleft = thrustleft +1;}
```

$a_{joystick\_y} = 0.1(\text{thrustleft})$

while (controller pulled back) {thrustback = thrustback +1;}

while (controller not pulled) {thrustback = 0;}

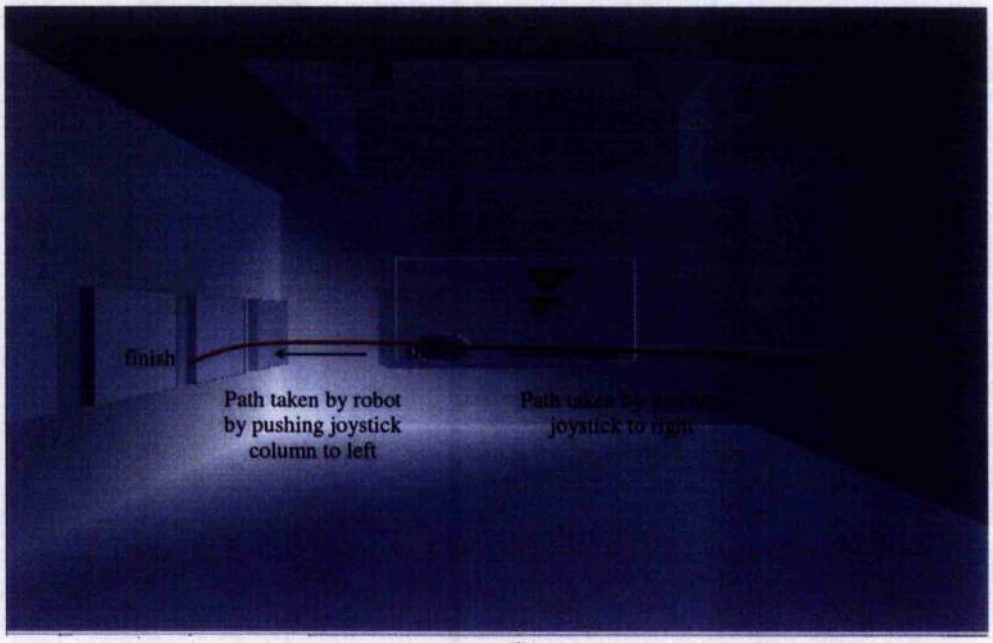
while (controller pulled forward) {thrustback = thrustback -1;}

$a_{joystick\_z} = 0.1(\text{thrustback})$

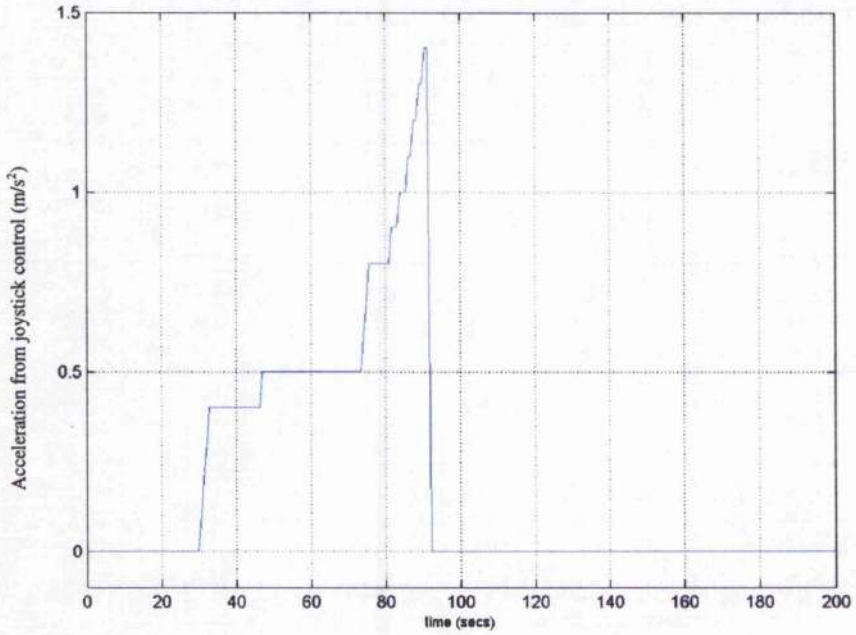
}

The joystick capabilities were initially tested in the 3-D environment before integrating with the other behaviours. **Figures 8.8a-e(i)** demonstrate the joystick competence by tracing the motion of the robot under the influence of *only* manual control. **Figures 8.8a(i)** shows the robot's trajectory when the joystick column is held to the right. Likewise, **Figures 8.8b(i)&c(i)** show the robot's trajectory when the joystick column is held forward and back, and finally, **Figures 8.8d(i)&e(i)** show the robot's trajectory when each of the two push buttons are held, which control the robot's acceleration along the z-axis. It should be noted that on release of the column, the acceleration drops to zero immediately.

The corresponding accelerations for each of **figures 8.8a-e(i)** over time is plotted in **figures 8.8a-e(ii)**, each demonstrating an increase in acceleration as the joystick position is maintained and a drop to zero when released. This is outlined in the pseudo-code, with the code adding an extra +1 or -1 to the thrust*value* as the joystick is held in each direction. This acceleration, in line with the other behaviours, is filtered when incorporated into the control system, to mimic pulse thruster firing.

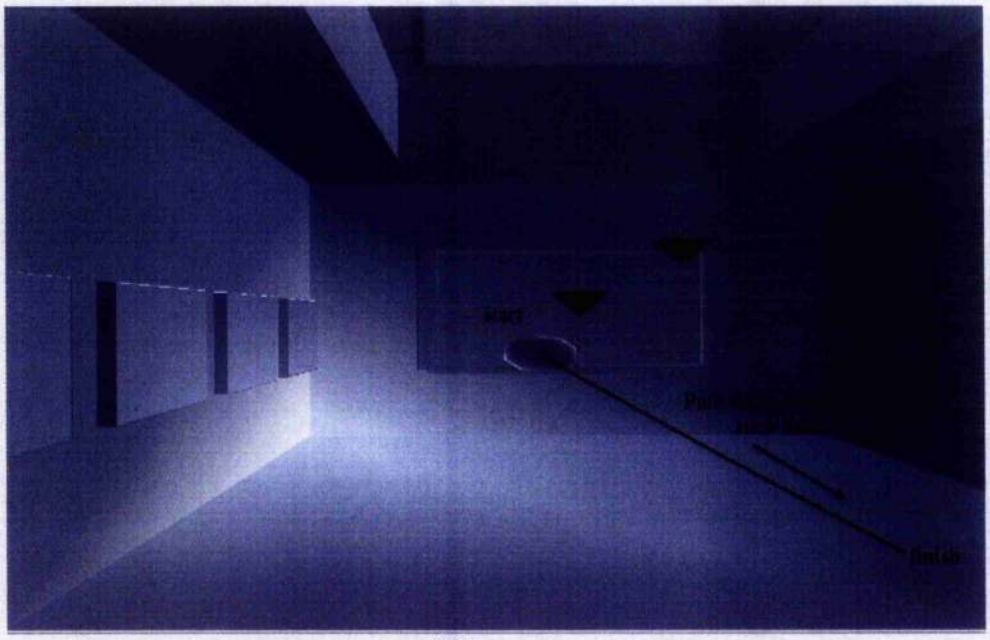


a(i)

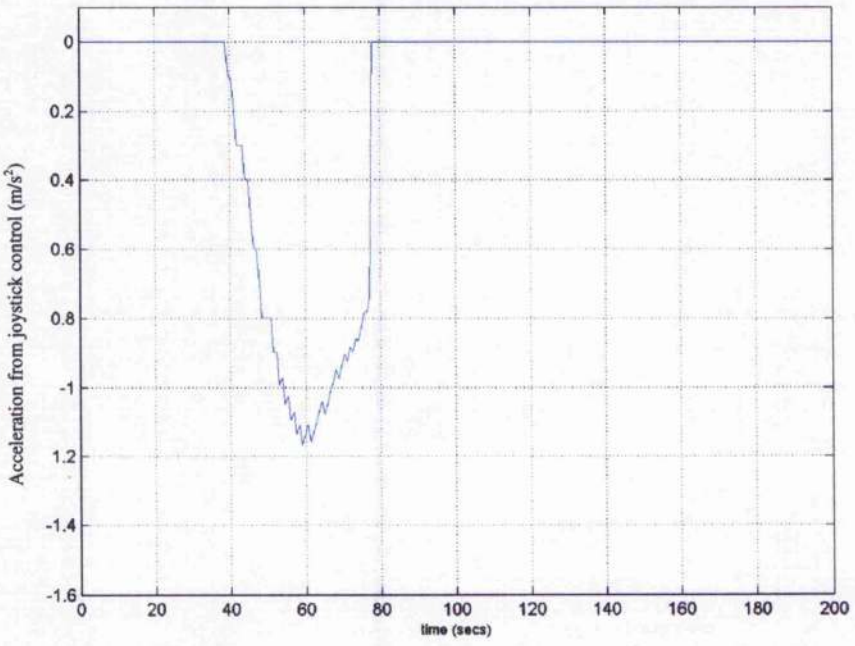


a(ii) right

Figure 8.8(a)-(e): Joystick Competence.

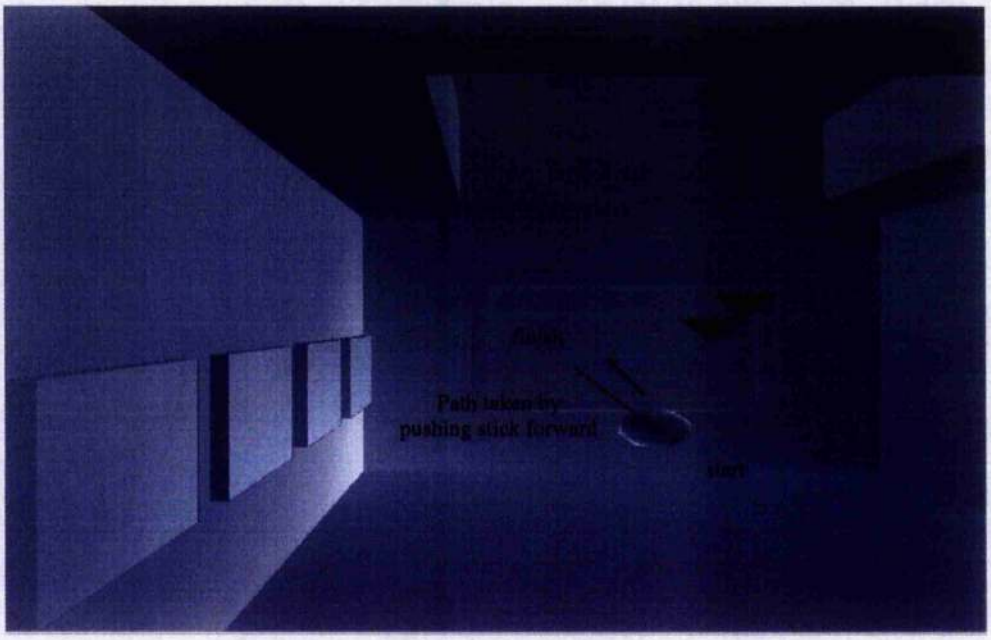


b(i)

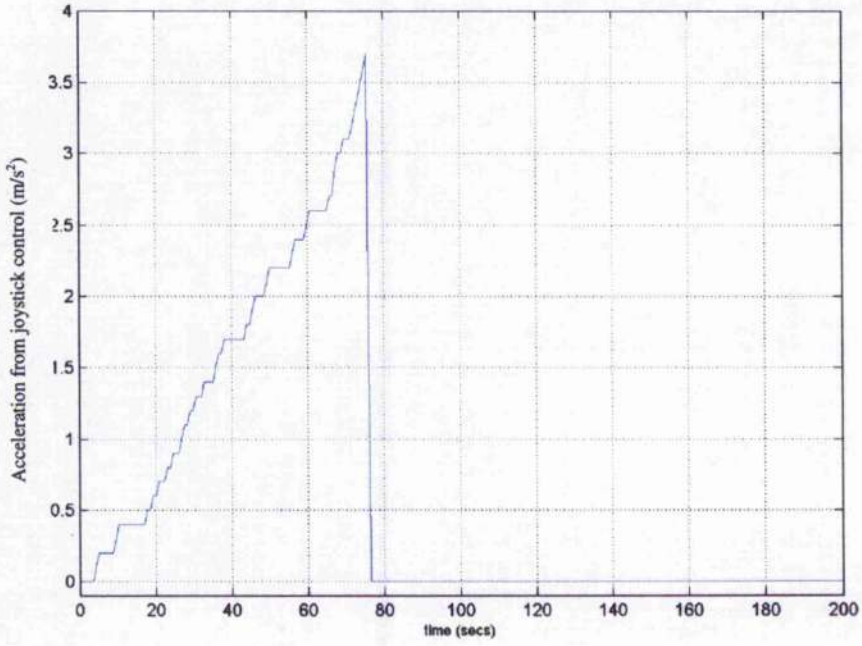


b(ii) back

Figure 8.8(a)-(e): Joystick Competence

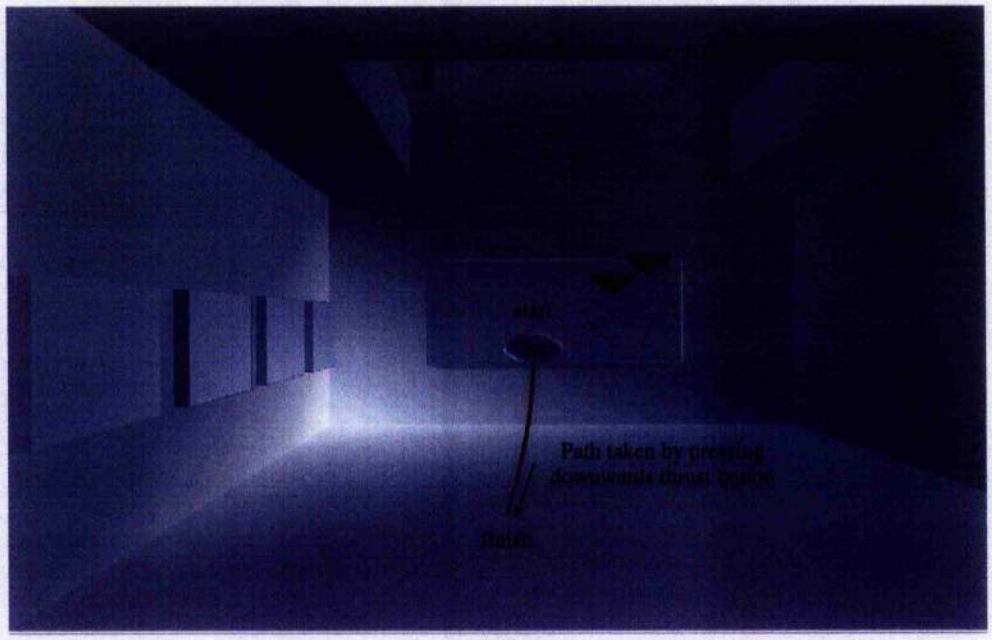


c(i)

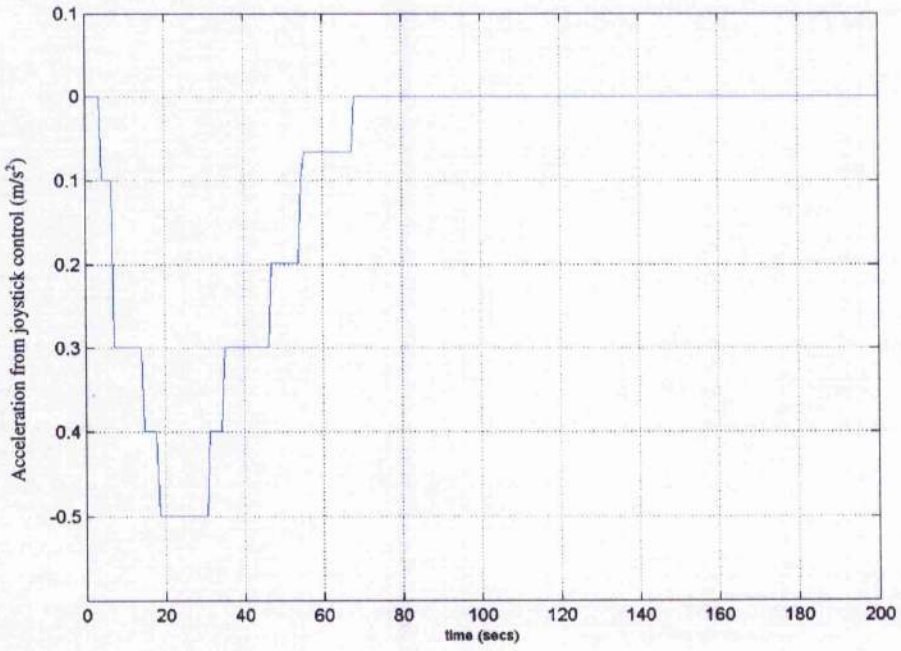


c(ii) forward

Figure 8.8(a)-(e): Joystick Competence.

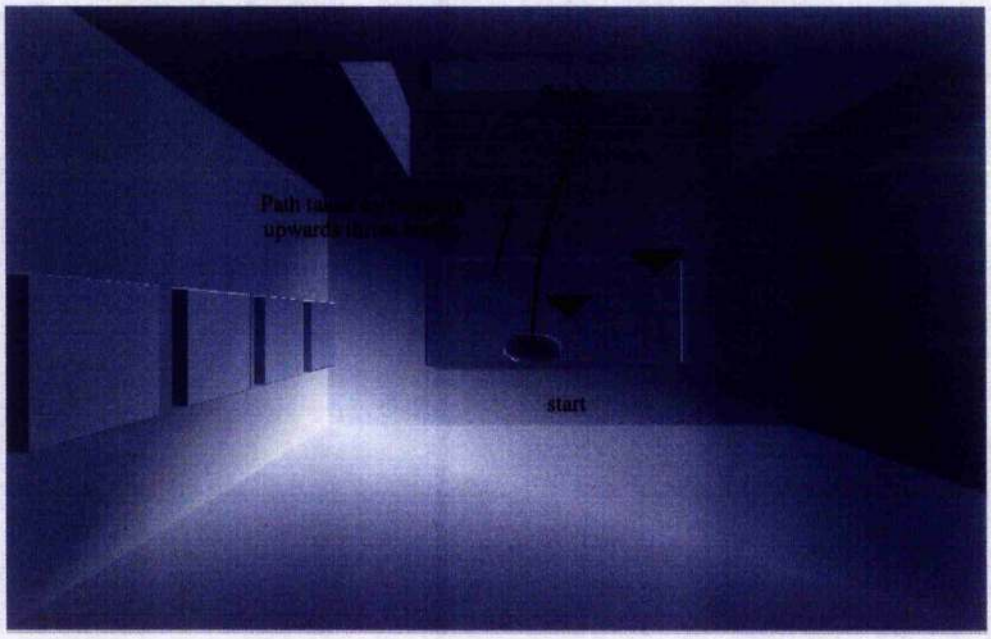


d(i)

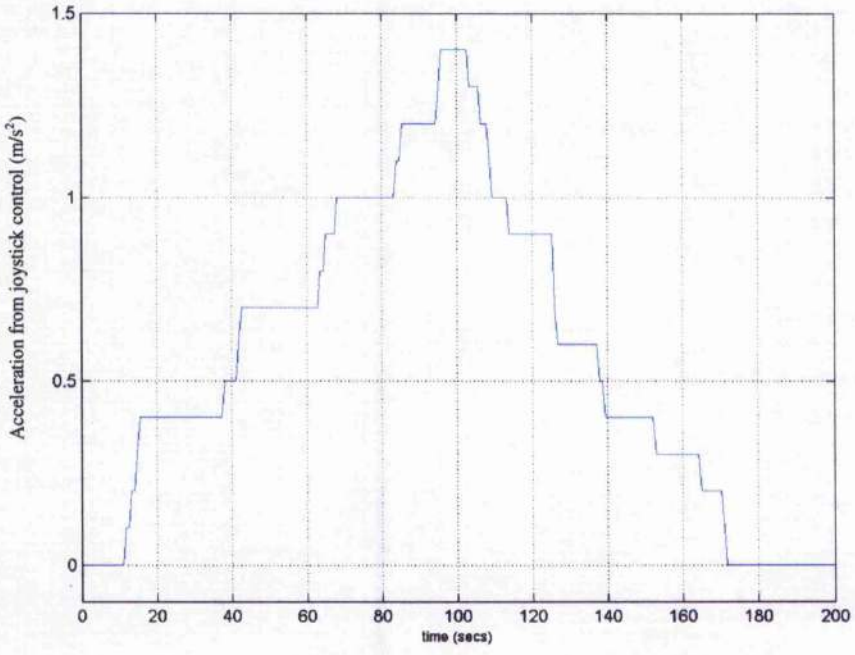


d(ii) down

Figure 8.8(a)-(e): Joystick Competence.



e(i)



e(ii) up

Figure 8.8(a)-(e): Joystick Competence.

It may be noted, however, from observation of **Figures 8.8a(i)-e(ii)** that if the control system can be governed purely by human control, with no autonomous intervention, then a collision may occur if the joystick is held for long enough. This may be avoided by incorporating human control into the integrated control system as a weighted behaviour, whereby obstacle avoidance strategies will dominate if human error allowed the robot to become in danger of collision. This allows un-trained operators to perform certain manoeuvres, and as the operator becomes more able this intervention could be gradually reduced. De-skilling of such complex tele-operation is of significant benefit for future deep space applications.

It is now intended to incorporate the human control algorithm into the model, weighted and summed like the previous behaviours. For the case of human control, the weighting is chosen to be directly proportional to the ratio of the distance to the nearest sensed obstacle to the sensor's obstacle detection range, such as in the case of the goal-searching and refuelling algorithms.

Such that:

$$\lambda_{human} = \frac{d_{near}}{range} \tag{8.10}$$

where λ_{human} = the joystick weighting.

From this, the final joystick acceleration becomes:

$$\underline{a}_{human\_final} = \lambda_{human}\,\underline{a}_{human} \tag{8.11}$$

where

$\underline{a}_{human}$ = joystick acceleration before weighting

$\underline{a}_{human\_final}$ = final joystick acceleration after the application of the weighting function.

This ratio is chosen such that as the distance to a detected obstacle is reduced, i.e., as human control of the robot hazards a possible collision by navigating close to

an obstacle, the weighting governing the human control acceleration is gradually reduced. This temporarily minimises the influence of this behaviour (a_{human_final}) on the chosen path of the robot, making way for, for example, collision avoidance strategies to dominate the robot's behaviour.

When the nearest sensed obstacle is detected just on the sensor's horizon, then the weighting becomes 1, and hence joystick control resumed fully, since it is decided the obstacle no longer posses a collision threat. Again, the ratio would never exceed one, since the robot would never detect an obstacle outside its sensor range.

Figure 8.9 was produced to demonstrate the variation in weighting of the human-input behaviour (on the x-axis) with distance detected to the nearest obstacle, over a period of 100 seconds. The vehicle, again equipped with a sensor object detection range of 3 metres, was released in an environment containing potential collision hazards. From the graph, it is noted that the variation in joystick weighting mirrors the distance detected to the nearest obstacle. If no obstacle can be detected, the sensors record a maximum value of 3 metres, the sensor horizon, as a default, and the joystick behaviour is maximised. However, as the robot approaches an obstacle within the sensor horizon, reducing distance to impact, the human-input weighting gradually decreases, de-rating this behaviour to make way for collision avoidance strategies. Likewise, as the robot retreats from the collision hazard, the joystick weighting gradually increases, allowing manual control to resume, having safely passed the collision hazard.

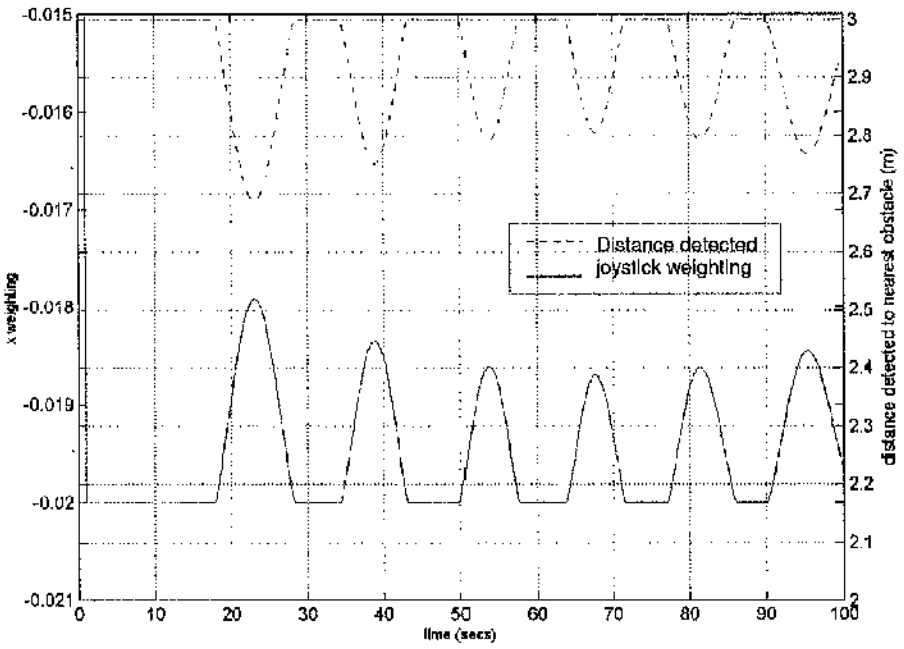


Figure 8.9: Plot showing the variation in joystick weighting with distance to nearest object.

The total acceleration for the robot now includes the human control. Such that:

$$\underline{a}_{total} = \underline{a}_{goal\_final} + \underline{a}_{avoid\_final} + \underline{a}_{refuel\_final} + \underline{a}_{human\_final} \tag{8.12}$$

i.e. $$\underline{a}_{total} = \lambda_{goal} \cdot \underline{a}_{goal} + \lambda_{avoid} \cdot \underline{a}_{avoid} + \lambda_{refuel} \cdot \underline{a}_{refuel} + \lambda_{human} \cdot \underline{a}_{human} \tag{8.13}$$

For the purposes of compatibility, a rule is included within the controller that will inhibit the behaviours of both the goal searching algorithm and the refuel algorithm in such instances when human control is active, incorporated using the following section of pseudo-code:

```
void compatibility()
    {
        if (λhuman >= 0)
            {
                λgoal = 0;
                λrefuel = 0;
            }
    }
```

However, the obstacle avoidance algorithm remains functional to maintain autonomous intervention in the event of human error causing a potential collision.

8.6 Scenario

The control methodology is now complete for the purpose of this thesis. This final section provides a scenario to demonstrate the capabilities of the working model.

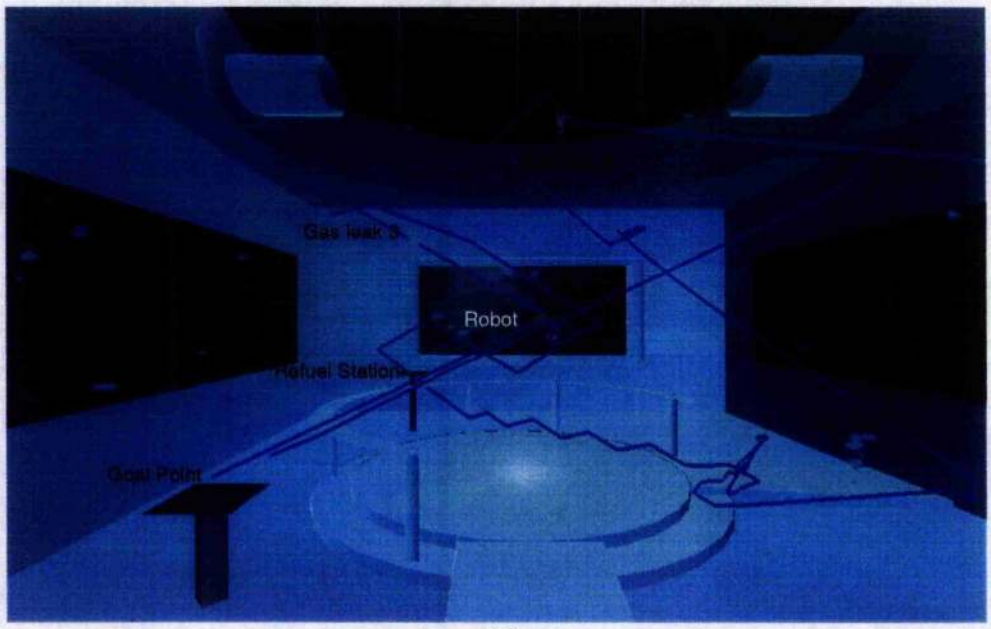
The robot is initially despatched from its start base, and asked to detect the source of three gas leaks placed at random positions within the module environment. The robot chooses the order in which to approach the leaks based on its cue-deficit capabilities, and must also, also refuel when opportunistic. When the task has been completed, the robot must finally return to its start base.

Figures 8.10(a) & **(b)** plot the paths taken by the robot during two separate runs of the task. The gas leaks are given different locations in **(a)** & **(b)** and are represented by a series of small spheres. The start base and the refuel base are also highlighted. In **figure 8.10(a)**, the robot initially heads to and reaches the closest gas leak (3). By reaching this gas leak, it is assumed that the leak is then sealed off (whether by summoning an astronaut, or by having high-level capabilities to achieve this autonomously), but by which, consequentially ends the concentration gradient

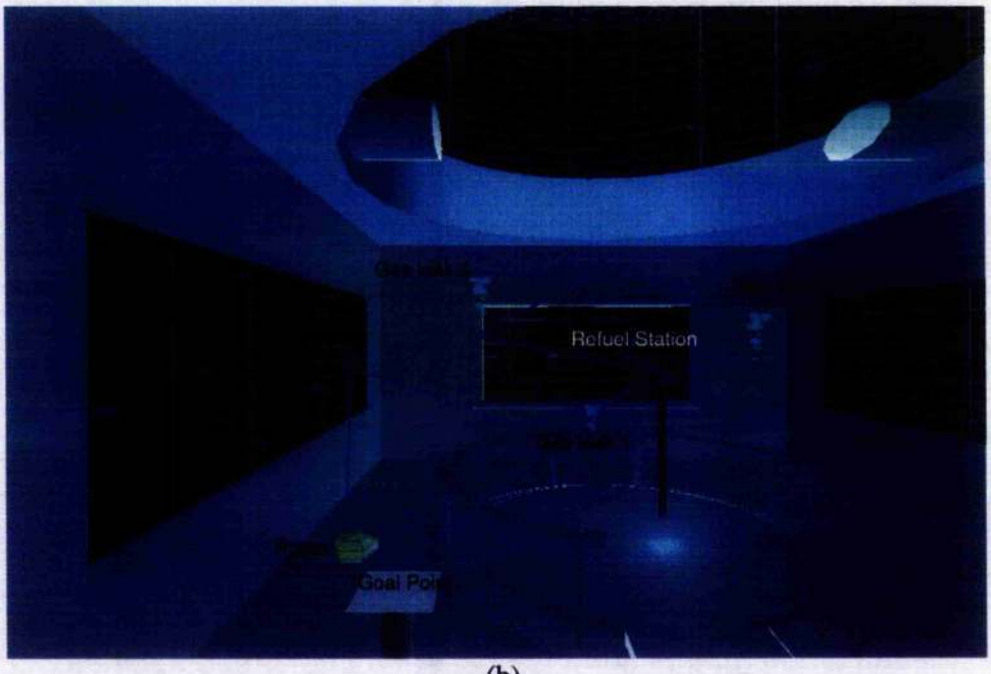
from this leak. Thereafter, having no further stimulus from this gas leak, it heads towards its next chosen gas leak (1) (by making a decision based upon its cue-deficit calculations) however, on passing close to the refuel station, the robot takes this opportunity to refuel before continuing to gas leak 1. After approaching and sealing off this gas leak, the robot then continues on to seek out and reach the final gas leak (2) before heading back to its goal base. In **figure 8.10(b)**, the robot likewise seeks out the most efficient route to reach all three randomly placed gas leaks whilst refuelling at opportunistic moments. In this example, the robot chooses to approach gas leak 1 first, then since it is passing the refuel depot on-route to gas leak 3, it takes the opportunity to refuel. It finally approaches gas leak 2 before heading back to the refuel station. These routes are different due to the difference in random positioning of the gas leaks which affect the choices made by the robot, since its decisions are based on the cue – deficit calculations which have thus been altered.

8.7 Conclusion

This chapter saw the development of a weighting function for each behaviour, enabling a seamless transition between one behaviour and the next, and improving the overall motion of the robot. Next, a final behaviour was incorporated, in the form of joystick control with force-feedback effects, providing the model with human intervention. This behaviour was again weighted and summed to the existing behaviours to provide human - robot synergy which allowed human intervention to inhibit other autonomous behaviours, such as goal searching and refuelling, but however allowing autonomous collision avoidance strategies to dominate if human error allowed the robot to become in danger of collision. This development allows a reduction in the operational complexity of overall control of the free-flyer, by deskilling the manual control and enabling the operator to proceed with more high level tasks.



(a)



(b)

Figures 8.10(a) & (b): Paths taken by the robot to complete the task shown in blue.

Chapter Nine: Conclusion

9.1 A Review

This thesis detailed the ongoing development of a useful robotic controller for free-flying space applications, allowing autonomous tasks to be successfully executed by providing the robot with decision making capabilities.

Initially, in **chapter 2**, a virtual environment was constructed to provide a test bed in which to test control algorithms and from which data may be extracted and analysed. The three dimensional environment was built using the OpenGl® software tool, and all algorithms and boundary conditions coded using the C programming language. The environment was designed with texture and lighting effects to resemble a space module, complete with potential collision hazards, a refuelling base and goal base. The robot was represented within the environment, and its design specifications established and integrated within the code.

In **chapter 3**, the development of a simple inertial control system was investigated and developed. The system was based on a theory by Braitenberg, in which a control method was adopted by direct interaction of the vehicle with its environment. Collision avoidance formed the initial architecture where the robot's virtual hardware was given proximity sensors to provide an estimation of the environment (lying along the sensor's line-of-sight and within their horizon), and provide a direct input to the robot's motion control. The collision avoidance acceleration was then filtered to mimic actual thruster activity by pulsing thrusters when a threshold value was reached.

The controller was expanded to incorporate a wall following algorithm to ease the robot's navigation through a constrained environment. This model was then

completed by the addition of the orbital dynamics of the robot relative to the orbiting spacecraft, in order to simulate the robot's slow drift motion.

In **chapter 4**, the acceleration control system developed in the previous chapter, was converted to a velocity control system, whereby the velocity magnitude remains constant, and instead, collisions were avoided by bending the velocity vector, manifest by appropriate thruster firing. This seen the model progress from the simple system in **chapter 3**, where the robot trajectory was translational, with the robot recoiling from obstacles elastically, to one seen in **chapter 4**, where the controller provided a smooth curved rotation from obstacles.

The control system was then capable of manoeuvring, collision free in an unknown environment. **Chapter 5** provided the robot with a useful function by developing a camera simulation tool, depicting the view seen through a virtual camera lens, and then integrating an attitude control system, developed using Lyapunov's method. This allowed the robot's optical axis to come to rest at any desired direction, or track a moving line-of-sight, and thus provided the free-flying robotic tool with a function as a camera aide whilst avoiding collisions.

In **chapter 6**, a guidance strategy was implemented based on a method used by biological systems called klinokinesis, to devise a path between an initial start point and finally goal destination. This method measures the signal strength emitted by the goal and a path is selected by following the concentration gradient. This goal tracking algorithm was weighted alongside the collision avoidance algorithm and integrated alongside the camera control torque, acceleration and speed controllers, providing a viable robot model with useful functionality.

In **chapter 7**, decision-making capabilities were incorporated to allow the robot to maintain self-sufficiency, whilst carrying out useful tasks. To be both useful and self-sufficient there must exist a trade-off between self-sustaining activities and work (which are often conflicting resources). In this chapter, a basic work cycle, consisting of: work - finding fuel - refuelling, was designed - energy was depleted when the robot was not refuelling, whilst the robot's utility was depleted when it was not working. A cue-deficit model was then developed to calculate the motivation (by

multiplying state variable deficit by resource cue) to carry out all possible behaviours, and by choosing that behaviour with the highest motivation. This allowed the robotic system to be self-sufficient, whilst carrying out useful work in the form of a series of chosen mission tasks. In **chapter 8**, the conversion from one behaviour to another was made a smooth transition by integrating a variable weighting, dependent on vehicle state as appropriately governed by the sensed environment, rather than an instant switch which is often rather clumsy.

In addition, a human controller was interfaced with the design, providing human-robot synergy to the now completed control system. This was incorporated in the form of a joystick, which allowed the operator to influence the overall behaviour of the robot from full autonomous control.

9.2 Future Work

The control methodology developed here is in its primitive stages. Whilst performing experiments in a virtual environment is fine as an initial test-bed, in reality, influences from sensor noise and imperfect thrusters may cause complications and errors in accuracy. Progression with this work further would benefit from developing the hardware and constructing a prototype free-flyer to test the methodology in a micro-gravity situation, producing accurate results and allowing for unforeseen problems to be addressed.

This piece of work is broad ranging, in that many different aspects of the development of the free-flying control system has been addressed in order to produce a completed system, however, with that, there has often been significant limitations and assumptions made along the way. For example, the robot has no navigation capabilities, in that it is not aware of its position in its environment, only what it senses directly, thus it is unable to navigate to a given point unless it is equipped with appropriate sensing tools to detect the destination point. The sensing gradient used in this work was based on a gas concentration gradient, which is only suitable for a few hypothetical situations, this limits the usefulness of the robot, and so more work could be done on extending its sensing capabilities, increasing its scope of use.

Collision avoidance strategies discussed herein, employ the use of acoustic sensors, which themselves have their limitations, whereby the beam width of the acoustic signal may be such that small obstacles are missed. There may also be interference where several robots are working within the same environment, or possible interference with onboard equipment. These limitations should be considered when choosing control hardware. Further to this, acoustic sensors are useless outside the space station due to the lack of atmosphere in space, and so other sensor techniques must be considered if the robot is to be used for EVA support.

The controller herein detects fuel dumps and goal points by following a simplified and uniform linear concentration gradient, this again is a simplification which could be explored further, to mimic more realistic activity.

This system uses compressed nitrogen gas thrusters; however, this method has the disadvantage of being more problematic and complex, compared to perhaps using electricity, and a propeller driven system. This or other alternative methods may be considered when choosing hardware components.

One further problem which could occur with the method employed here could arise in attitude control with there being the possible occurrence of singularities arising whilst bringing the camera to rest at a desired orientation. For the purpose of this investigation singular orientation is avoided by choosing suitable target pointing angles, however, a flight vehicle would require developing quaternion-based calculations to avoid the possibility of loss of stability of the camera orientation system.

Although there are these limitations, the intention here was to present a *complete* system based on a simple reactionary robotic control, which has been accomplished.

Integrating this work, the control methodology could include incorporating a deliberative path planner, to navigate around a known environment, as a top level agent, with the reactive modes developed in this investigation coming in to place to deal with unexpected obstacles. Integrating the local collision avoidance

strategies with a path planner using potential functions would increase the robots efficiency and usefulness, by allowing path constrained proximity manoeuvring for space station applications despite unpredicted changes to the environment. Such uses of these algorithms for the autonomous control of free-flying vehicles can plan collision-free paths to reach and retreat from observation points around space structures whilst incorporating reactive, real-time, behaviour based control in unexpected or dynamic situations, enabling collision-free trajectories both exterior to and interior to large space structures such as the International space station.

With a single robot model complete, future work may progress to look at the possible development of co-operation between numerous robots, forming a *multi-agent system*. A *multi-agent system*, where robots co-exist and function as part of a larger community, would be capable of performing autonomous activities, however, they would mutually influence each other through sharing the same environment and resources, and would co-operate to provide an advantage to the group (Wooldridge, 1997), (Doran, 1996), (Lesser, 1999), (Boella, 1999). The benefits of multi-agent systems are increased flexibility and the capability of approaching a wide variety of applications or widely distributed resources, providing "*a net effect greater than the sum of the parts*" (Hayes, 1999).

A scenario is proposed here for use of a possible future development of a multi-agent system composed of an Environmental Control System (ECS), a free-flyer and a crew member, working to maintain safe environmental conditions for life support onboard the International Space station. Initially, the ECS detects a dangerous level of CO_2 present in one of the station modules. To diagnose the problem, it must determine if the problem lies in a faulty sensor, or is indeed a gas leak. It initially commands the free-flyer to navigate to the module where the gas leak has been detected. The free-flyer uses a top level agent path-planning tool which uses potential functions and a pre-determined knowledge of the space station, to navigate to the module, whilst incorporating reactive, real-time, behaviour based control to avoid unexpected collisions, or to use opportunistic refuelling strategies, if necessary. When the free-flyer reaches the module, it records the gas concentration. There are two possible scenarios:-

1) The gas concentration measured by the free-flyer is nominal, differing from that recorded by the ECS. In this situation, the ECS then concludes that the sensor is faulty and by means of a user interface (U.I.) commands the crew member to replace the sensor. The ECS commands the free flyer to station-keep at the module until the crew member enters the module, and then to follow the crew member whilst he undergoes the repair in order to provide assistance by means of a communication link, if necessary. When the sensor has been replaced, the crew member confirms this to the ECS by means of the U.I. and on confirmation, the ECS records the gas concentration, which is now found to be nominal. The ECS then commands the free-flyer to return to its station, having successfully completed its task.

2) The gas concentration recorded by the free-flyer is found to be the same as that detected by the ECS. In this scenario, the ECS infers that there is a gas leak and commands the free-flyer to source the position of the leak. The free-flyer determines the position of the leak by the methodology employed herein, which then allows the ECS to command that portion of the module to be sealed off. The crew member is informed of the leak via the UI, who then carries out the necessary procedures to ensure the problem is fixed. He confirms this to the ECS, which then determines the new concentration level, found to be nominal. The ECS further commands the free-flyer to record the new concentration level, which is also found to be nominal, and so the free-flyer returns to its station.

The benefits of behaviour based reactive control in an autonomous agent also have implications in deep space missions, such as the exploration of unknown, hostile or unpredictable environments such as Mars exploration. A multi-agent system in this situation would be advantageous as the risk of mission failure would be reduced if

swarms of micro- or nano- robots were released rather than having only a single, larger rover. The shared goal within the multi-agent system would provide robust working systems more capable of survival in unpredictable environments since it would be less of a cost to lose part of the colony through defects than if there were only one robot. Future work could look into developing a colony of simple control methodologies which would share the same environment, resources and goals, distributing the workload to produce efficient results as outlined in the scenarios above.

References

Boella, G., Damiano, R. & Lesmo, L., "*Cooperation and group utility. In Intelligent Agents VI --- Proceedings of the Sixth International Workshop on Agent Theories, Architectures, and Languages (ATAL-99, Orlando)*",Berlin, pages 319-333, 2000.

Beer, R., "Intelligence as Adaptive Behaviour. An Experiment in Computational Neuroethology" Academic Press, 1990.

Birk, A., "Behaviour Based Robotics, its scope and its prospects", 24th annual conference of IEEE Industrial Electronics Society, 1998.

Bluethmann, W., Ambrose, R., Diftler, M., Askew, S., Huber, E., Goza, M., Rehnmark, F., Lovchik, C., & Magruder, D., "Autonomous Robots", Springer Science+Business Media B.V., Vol 14, pp 179 – 197, 2003.

Boella, G., Damiano R., Lesmo L., "A Utility Based Approach to Co-operation among Agents" In van der Hoek, W., Meyer, J.-J., and Witteveen, C., editors, ESSLLI99 workshop: Foundations and applications of collective agent based systems (CABS), 1999.

Bradshaw, M., Sierhuis, M., Gawdiak Y., "Human-Centered Design for the Personal Satellite Assistant," In International Conference on Human-Computer Interaction in Aeronautics 2000, Toulouse, France.

Braitenberg, V., "Vehicles, Experiments in Synthetic Psychology," MIT Press, Cambridge, Massachusetts, 1984.

Brooks, R., "AI through Building Robots," Technical Report A.I. Memo 899, Massachusetts Institute of Technology, May 1986.

Brooks, R.A., "Elephants Don't Play Chess," Robotics and Autonomous Systems 6. MIT Artificial Intelligence Laboratory, Cambridge, 1990.

Brooks, R.A., Aryananda, L., Edsinger, A., Fitzpatrick, P., Kemp, C.,. O'Reilly, M., Torres-Jara, E., Varshavskaya, P., and Weber, J., *"Sensing and Manipulating Built-For-Human Environments"*, International Journal of Humanoid Robotics, 1:1, March 2004, pp. 1–28.

Carde, R., & Bell, W., "Chemical Ecology of Insects," Chapman Hall, 1984.

Chobotov, A., "Orbital Mechanics," American Institute of Aeronautics and Astronautics, Washington, 1991.

Choset H., Kortenkamp D., "Path Planning and Control for AERCam, a Free-flying Inspection Robot in Space," ASCE Journal of Aerospace Engineering, Proceedings of ICRA '99, Vol. 2, pp. 1396 – 1403, 1999.

Darwin, C., "The Origin of the Species," John Murray, Albemarle Street, 1859.

Dennett, D., "Intentional Systems in Cognitive Ethology" Behavioural and Brain Sciences 6, p343-390, 1983.

Dennett, D., "Why Not the Whole Iguana?" Behavioural & Brain Sciences 1: 103 - 104, 1978.

Doran, J., Franklin, S., Jennings N., Norman, T., "On Co-operation in Multi-Agent Systems" UK Workshop on Foundations of Multi-Agent Systems, Warwick, 1996.

Dorias, G., and Nicewarer, A.,"Adjustably Autonomous Multi-Agent Plan Execution with an International Spacecraft Free-Flying Robot Prototype", Proceedings of the 13 the International Conference on Automated Planning, 2003.

Doty, K. & Bou-Ghannam, A., "Controlling Situated Agent Behaviours with Perception and Cognition," IBM, Florida, 1994.

Dunham, W., "Euler: The Master of Us All", Washington: Mathematical Association of America 1999.

Euler, L., "Introductio in analysin infinitorum" English translation *Introduction to Analysis of the Infinite* by John Blanton Springer-Verlag 1988.

Flynn, A.M. "Gnat Robots (And How They Will Change Robotics)", Proceedings of the IEEE Micro Robots and Teleoperators Workshop, 1987.

Foley T.M., Engineering the Space Station, Aerospace America, pp. 26-32, October 1996.

Fortescue, P., Stark, J., "Spacecraft Systems Engineering," Wiley & Sons Ltd., West Sussex, 1991.

Glenmar A., "An Attitude Stabilisation Device for Free-Floating Tools." MSc Degree Project Report, Vehicle Engineering Program, Department of Mechanics, The Royal Institute of Technology, Sweden, 1999.

Gillies E., Johnston A.G.Y., McInnes C.R., "Action Selection Algorithm for Autonomous Micro-spacecraft", Journal of Guidance, Navigation and Control, Vol. 22, pp. 914-916, 1999.

Gillies, E., McInnes, C., & Neil, D., "Emergent Search Behaviour in a Sensory Gradient," Journal of Guidance, Navigation and Control, 1994.

Girard, B., Filliat, D., Meyer, J.A., Guillot A., "Integration of Navigation and Action Selection Functionalities in a Computational Model of Cortico-Basal-Ganglia-Thalamo-Cortical Loops", Adaptive Behavior - Animals, Animats, Software Agents, Robots, Adaptive Systems, Vol. 13, pp. 115 – 130, Issue 2, June 2005.

Grey-Walter, W., "The Living Brain," Gerald Duckworth and Co Ltd, 1953.

Hayes, Caroline., "Agents in a Nutshell- A Very Brief Introduction" IEEE Transactions on Knowledge and Data Engineering Vol. 11, No. 1, January-February 1999.

James, G., "Modern Engineering Mathematics", Addison-Wesley, 1994.

Kalman R.E., Bertram J.E., "Control System Analysis & Design via the Second Method of Lyapunov Part 1: Continuous Systems," ASME, Vol.82, pp. 371-393, June 1960.

Kalman R.E., Bertram J.E., "Control System Analysis & Design via the Second Method of Lyapunov Part 2: Discrete Systems," ASME, Vol.82, pp. 394-400, June 1960.

Khatib, O., Brock, O., Chang, K., Ruspini, D., Sentis, L., Viji, S., "Tracts in Advanced Robotics,", Springer-Verlag, Vol 6, pp. 239 – 254, 2003.

Lesser, Victor., "Co-operative Multi-Agent Systems: A Personal View of the State of the Art" IEEE Transactions on Knowledge and Data Engineering Vol. 11, 1999.

Liu, J. & Khatib, O., "Practical Connection between Potential Fields and Neural Networks," Proceedings of the Workshop on Intelligent Manipulation for Manufacturing Automation, Hong Kong, 1999.

Lyapunov, A.M., "The General Problem of the Stability of Motion," Taylor & Francis, 1992.

Maes, P., "Modeling Adaptive Autonomous Agents," Artificial Life Journal, Vol. 1, Massachusetts Institute of Technology, pp. 135 - 162, 1994.

Maes, P., "A bottom-up mechanism for behaviour selection in an artificial creature," Proceedings of the First International Conference on Simulation of Adaptive Behaviour, MIT Press: Cambridge, MA, 1991.

Menzies, E., Das, N. & Wood, D., "Behaviors Producing Photodispersal in Stentor Coeruleus," American Society for Photobiology Journal, preprint:N/A–N/A (2006)

Meriam, J. & Kraig, L., "Engineering Mecahnics - Dynamics", Volume II, Third Editin, John Wiley & Sons, 1993.

Meyer, R., "Elements of Space Technology," Academic Press, 1999, pp13-14.

Moravec, H., "Locomotion, Vision and Intelligence, in Robotics Research 1," Brady and Paul, MIT Press, 1984.

Murphy, J., "Advanced Calculus for Engineering and Science Students", Arklay Publishers, 1993.

McFarland, D.J & Houston, A.I, "*Quantitative Ethology - The State Space Approach* " Pitman, London, 1981.

McFarland, D.J & Spier, E., "*Basic Cycles, Utility and Opportunism in Self-sufficient Mobile Robots*" Robotics and Autonomous Systems, 20:179-190, 1997.

McQuade F., "Autonomous Control for On-Orbit Assembly Using Artificial Potential Functions," PhD thesis, University of Glasgow, 1997.

Nerem, S., "Spaceflight Dynamics – Clohessy Wiltshire (CW) Equations", University of Colorado, Boulder, lecture 12, November 2004.

Patten, L., "Innovative Enhancements for Reducing the Crew Time Needed for On-Orbit Robotic Maintenance Operations on the International Space Station" 54th International Astronautical Congress of the International Astronautical Federation, the International Academy of Astronautics, and the International Institute of Space Law, Bremen, , Sep. 29-3, 2003.

Press, W.T., Teukolsky, S.A., Vetterling, W.T. & Flannery, B. P., "The Art of Scientific Computing" Cambridge University Press, 2002, pp710 – 714.

Radice, G., Gillies, E.A. & McInnes C. R., "*Cost Function Analysis for Autonomous Clustered Microspacecraft*", American Institute of Aeronautics, 2000.

Resnick, M., "Lego, Logo and Life," Proceedings of the interdisciplinary workshop on the synthesis & simulation of living systems. SFI studies in the sciences of complexity, Vol. 6, Christopher Langton editor, Addison-Wesley, Redwood City, CA p397-406, 1989.

Reynolds. C.W., "Interaction with groups of Autonomous Characters," In proceedings of Games Developers Conference, 2000, Miller Freeman Game Group, San Francesisco, CA, 2000.

Roderick, S., Roberts, B., Atkins, E., Churchill, P. & Akin, D., "An Autonomous Software Safety System for a Dexterous Space Robot," Journal of Aerospace Computing, Information, and Communication 2004 1542-9423 vol.1 no.12 pp 564-579.

Roger A. B., M^{c}Innes C. R., "Passive-Safety Constrained Free-Flyer Path Planning with Laplace Potential Guidance at the International Space Station," Journal of Guidance Control & Dynamics, vol 23 no. 6, pp 971 – 979.

Rouche, N., Habets, P. & Laloy, M., "Stability Theory by Liapunov's Direct Method,", Springer-Verlag, New York, 1977.

Scriber, JM., Bell, WJ. & R Cardc "Chemical ecology of insects", Chapman and Hall, 1984.

Sexton, C., "C Programming Madc Simplc," Butter-worth-Heinemann, Reed Educational and Professional Publishing, 1997.

Sibly, R.M. & McFarland, D.J., "*A State Space Approach to Motivation*" Motivational Control Systems Analysis, Academic Press, London 1974.

Sibly, R.M. & McFarland, D.J., " *On the Fitness of Behavior Sequences*" American Naturalist, Vol. 110, No. 974), pp. 601-617, Jul. - Aug., 1976.

Sibly, R.M. & McFarland, D.J., " *The Behavioural Final Common Path*" Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences, Vol. 270, No. 907 pp. 265-293, May 15, 1975.

Sidi, M. J., "Spacecraft Dynamics & Control," Cambridge University Press, Cambridge, 1997.

Sierhuis, M.; Bradshaw, J.M.; Acquisti, A.; Hoof, R.v.; Jeffers, R.; and Uszok, A. "Human-Agent Teamwork and Adjustable Autonomy in Practice," in Proceedings of The 7th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS), Nara, Japan, 2003.

Simmons, J., "The giant book of scientists: The 100 greatest minds of all time", Sydney: The Book Company 1996.

Spier, E. & McFarland, D.J, "*Possibly Optimal Decision Making Under Self-sufficiency and Autonomy*" Journal of Theoretical Biology, in press, 1996 (a).

Spier, E. & McFarland, D.J, "*A Finer-Grained Motivational Model of Behaviour Sequencing*" Journal of Theoretical Biology, in press, 1996 (b).

Steels, L., "A Case Study in the Behaviour-Orientated Design of Autonomous Agents," submitted to SAB-94, 1994(a).

Steels , L., "Mathematical Analysis of Behavioural Systems," 1994(b).

Thomson, W., "Introduction to Spacecraft Dynamics," Dover Publications, New York, 1986.

Travers, M., "Animal Construction Kits," In Artificial Life, ed. C. Langton. Addison-Wesley, 1988.

Volpe, R. & Ivlev, R., "A Survey and Experimental Evaluation of Proximity Sensors for Space Robotics," Proceedings of the IEEE International Conference on Robotics and Automation, May 8-13 1994, San Diego.

W Grey Walter – "*The Living Brain,*"Gerald Duckworth and Co., Ltd, 1953.

Welsh, T., "Automated Microsatellite Model Using Simulink®," 4th Year Project, Department of Aerospace Engineering, University of Glasgow, 1999.

Wertz, J.R., "Space Mission Analysis and Design," 2nd Edition, Kluwer, Dordrecht, 1991.

Wie, B., "Space Vehicle Dynamics and Control" American Institute of Aeronautics and Astronautics, Inc., Reston, Virginia, 1998.

Wiesel, W. E., "Spaceflight Dynamics" (Second Edition), McGraw-Hill, 1997.

Wilde, D., Sytin, O., "The Mir-Progress-Inspector Mission," International Symposium Space Dynamics, 1995.

Wilson, S., "*The Animat Path to AI,*" Proceedings of the First International Conference on the Simulation of Adaptive Behaviour, Cambridge, Massachusetts: The MIT Press/Bradford Books 1991 (pp 15-21).

Whyte, V.,"Free-Flyer Path Planning using Artificial Potential Functions," Department of Aerospace Engineering, University of Glasgow, 1998.

Woo, M., Neider, J., Davis, T. & Shreiner, D., "OpenGl® Programming Guide," Third Edition, The Official Guide to Learning OpenGl®, Version 1.2, Addison Wesley Longman, 1999.

Wooldridge, Michael., "*Agent-Based Software Engineering*" Mitsubishi Electric Digital Library Group, London, 1997.

Website References:

AERCam, http://spaceflight.nasa.gov/station/assembly/sprint/ (last visited 12/03/05)

Aercam Objectives, http://www.tvcameramen.com/equipment/nasacam.htm (last visited 01/02/06)

Aleksandr Mikhailovich Lyapunov, http://www-history.mcs.st-andrews.ac.uk/Mathematicians/Lyapunov.html (last visited 12/04/05)

A New Generation of Space Robots, http://spaceflight.nasa.gov/station/eva/robotics.html (last visited 12/04/05)

Canadarm, http://www.space.gc.ca/csa_sectors/human_presence/canadarm.html (last visited 07/03/05)

Canadarm2, http://www.mdrobotics.ca/canarm2_frame.html (last visited 12/03/05)

International Space Station, http://www.boeing.com/defense-space/space/spacestation/flash.html (last visited 12/07/05)

Inspector, http://www.eads.net/cads/en/index.html (last visited 02/06/05)

ISS Overview, http://www.eads.net/eads/en/index.html (last visited 22/04/05)

NASA News, http://amesnews.arc.nasa.gov/releases/1999/99_53AR.html
Personal Satellite Assistant, http://ic-www.arc.nasa.gov/ic/psa.html (last visited 12/02/04)

Space Station Assembly, http://spaceflight.nasa.gov/station/assembly/index.html (last visited 01/06/05)

Space station EVA, http://spaceflight.nasa.gov/station/eva/index.html (last visited 23/01/06)

Space Station History, http://spaceflight.nasa.gov/history/station/index.html (last visited 12/08/04)

SPDM,
http://www.space.gc.ca/csa_sectors/human_presence/iss/contribut/mss/spdm.html (last visited 17/04/05)

The Shuttle Remote Manipulator System,
http://ewh.ieee.org/reg/7/millennium/canadarm/canadarm_technical.html (last visited 05/09/04)

Welotec GmbH – Partner for automation applications – sensors, light barrier, radio data transmission, http://www.welotec.de/index last visited November 2003.