Reverse Engineering an Active Eye

Hanson Schmidt-Cornelius

Submitted for the degree of D. Phil.

University of Sussex

15th December 2002

Declaration

I hereby declare that this thesis has not been submitted, either in the same or different form, to this or any other university for a degree.

Signature:

Acknowledgements

I am very grateful to Hilary Buxton and David Young, who provided outstanding support and ideas during and outside of numerous meetings. The working environment was very encouraging, enthusiastic and constructively critical, which took the project and its experiments far.

Special thanks also to Phil Husbands on the research committee, who always supported the ideas of this research, even during times of radical conceptual changes.

I am also most grateful to Ben du Boulay, who influenced my career path by encouraging me to explore my academic potential.

Many thanks also to Robert Woodhouse and Brian Williams, who contributed vital engineering skills and workshop resources to help design and build mechanical components for the experimental environment.

A special thank you to the COGS support team, for maintaining the software. In particular Simon Nichols and Richard Grainger, who supported the very specific software configuration of the experimental system.

I would also like to thank Dorothea Vehling-Schmidt and my parents for their time, moral and financial support throughout the research period.

Finally I would like to thank my wonderful wife Mei-Ling, for her support, patience and unconditional love. She stood by me throughout my work and did not get bored with me, even during the long times we were apart.

Reverse Engineering an Active Eye

Hanson Schmidt-Cornelius

Summary

The research presented in this dissertation investigates new concepts of biologically inspired active vision systems, using high power, linear actuators. First, fundamental vision paradigms are discussed, contrasting passive and purposive active vision. This leads to physiological and historical investigations into extra-oculomotor mechanisms and laws that define human eyeball positioning. Based on this background, the design of a monocular extra-oculomotor robot is devised that combines a parallel architecture, static balancing and opposing actuation. This design is then implemented, resulting in the construction of an experimental active eye environment. The mechanical, electrical and software development phases are discussed.

The unique control and mechanical properties of the active vision system here necessitate a range of low level control algorithms that are normally not required for conventional vision systems with stepper, servo or direct drive motor control. Three biologically based control models are used to form the basis of a range of adaptive control models that are tested on the experimental environment of the mechanical eye. The controllers developed here perform similar operations to those expected from the control layers of biological vision systems. This is especially the case for the neural control model of the superior colliculus, which has not been tested in the context of such an environment before.

Experimental results show that this new approach to artificial active vision systems is viable, replacing position controlled actuation with dynamic mechanisms, regulated by adaptive controllers.

Submitted for the degree of D. Phil.

University of Sussex

15th December 2002

Contents

List of Figures

List of Tables

Chapter 1

Introduction

To most people, the *eye* would mean a human eye, and in the context of this dissertation, references to biological eyes will mean a human or other mammalian eye. However, it has to be stressed that there is a rich diversity of natural eyes with many different architectures. It is outside the scope of this research to cover these eye systems, but Land and Nilsson [70] have published a comprehensive introduction to this field.

Mammalian eyes can be pointed in different directions by contractions of a number of muscles. In humans one distinguishes between two main types of eye movement strategies: saccades and smooth pursuit. Investigations into the behaviour and control of these movements have utilised a wide range of techniques, such as: the observation of controlled eye movement tasks, the measurement of brain activities, the simulation of possible eye positioning control strategies and the development of physical active vision systems on which eye control strategies can be tested. As a result, the physical behaviour of the extra-oculomotor system has been fairly well specified but the control structures that underlie the control of this system remain relatively unexplained.

1.1 Orientation

The contents of this dissertation are unashamedly diverse, due to the complex nature of active vision research. During the investigation of artificial active vision, much research has focused on the development of gaze control algorithms that control artificial active vision systems. The interface between gaze control and active vision platforms has usually been fairly straightforward, as highly engineered mechanics allow a high degree of feedback independent repeatability and positioning precision. Although such systems can show minor signs of fatigue and performance degradation, the effects are often negligible and not nearly as dramatic as the degree of fatigue that can be experienced by biological systems. The extra-oculomotor system, in particular, fatigues very quickly and recovers again, resulting in constant performance fluctuations. Biological vision systems are, however, able to adapt to these changes, without individuals being significantly aware of fatigue or a change in performance.

It is a fact of research that not all artificial active vision systems are designed to simulate aspects of biological vision. Even in the cases where biological vision is to be simulated, it is

often questionable to what degree artificial vision platforms really do represent good models of biological systems. This problem is not so much apparent with high level gaze control strategies for smooth pursuit or saccadic movements, but with low level control mechanisms that are necessary to interface between gaze control and the vision hardware. In most artificial vision platforms, gaze control instructions can normally be directly converted into motor control commands that reliably place vision systems into a predictable position. As the human extra-oculomotor system has fluctuating positioning properties, high level gaze control either needs to be able to adapt to the dynamics of these changing properties or interface with low level control mechanisms that provide a predictable position control to higher level control mechanisms. This may appear to be a subtle problem, but it can have a significant impact on the control of active systems. The widely accepted use of highly engineered systems puts high requirements on the electrical and mechanical properties of active vision systems, but neglects the fact that software can be implemented to interact dynamically in changing situations.

1.2 Aims and Motivation

The research aims to explore aspects involved in designing and building artificial active vision systems with changing positioning accuracy. For this, it is necessary to approach artificial active vision from an entirely new direction and rethink current design philosophies. This will inevitably raise the performance requirements of other components within the active vision system. In particular, there will be greater demands and new requirements on adaptive control issues that are involved in operating such systems. For example, the question of feedback requirements will arise. Is feedback required? If so, what type of feedback? This investigation is undoubtedly a truly interdisciplinary challenge but necessary to convey the potential problems and capabilities that are expected to be present in a mechanically changing system. This work is expected to contribute novel design paradigms to the field of artificial active vision, both on the hardware and software level. Results are also expected to show that such alternative approaches to active vision can result in systems that use less accurate hardware, are cheaper to build and possibly just as reliable as conventional active vision systems. The following points list the main motivations behind the work conducted here:

- One of the most successful active vision systems in the world, the human extra-oculomotor system, uses a fatiguing and recovering vision platform. It may be possible to use observations of such a system to design and build an artificial vision platform with similar properties.
- If it is possible to develop an artificial vision system with cheaper and less reliable components that have biologically plausible properties, like the ability to fatigue and still be similarly or equally effective as systems developed to high engineering standards, this may mark the beginning of a new generation of artificial vision systems. This may be of particular interest for high volume systems with short life cycles.
- The control issues involved in operating low accuracy active vision systems have been shown to work in biological vision, but certain issues have not yet been explored in the control of artificial active vision systems. This could open a new approach to active vision control and further the understanding of biological control concepts within the human oculo-motor system.

These are indeed very motivating new areas that could revolutionise the development of active vision platforms in the future and further existing knowledge about biological control systems.

1.3 Key Concepts

One of the primary advantages of developing a physical artificial active vision platform is that it can interact with the real world. For the application of active vision, this is a particularly important factor. It may be possible to simulate many aspects of a biologically inspired active vision system, but one would certainly run into problems no later than when an attempt was made to test real world images, requiring interaction with the system. During the development and the investigation into building the vision system, it is necessary to focus on key issues within the research field of active vision. As the artificial active vision platform designed here is inspired by nature, biological plausibility must be a key point from which to develop the new system. This raises the necessity to find a means by which observed properties of the human extra-oculomotor system can be translated into design requirements for a new vision system. The system cost will also play a significant role as the budget is limited. This also links back to the motivation of developing a cheap active vision system with good performance capabilities. As the system design will be heavily based on anatomical findings, it may be envisaged that the electrical control should also use a biologically plausible approach. The software control could also take account of this fact. For some control paradigms it may even be easier if activation-potential control mechanisms existed in the system. Some interesting control concepts originating from biological research may literally just “plug” into the active vision system.

The following list outlines the stages of the research:

- Investigate what anatomical properties of the human extra-oculomotor system could be utilised to develop an artificial active vision platform. These properties should then form the foundation on which to base the requirements specification for the artificial vision system.
- Design and build an artificial active vision platform, based on the requirements specification extracted from the anatomical properties of the human extra-oculomotor system.
- Develop a targeting system that produces unambiguous, visual target patterns, which can be controlled precisely and executed repeatedly. This will allow the objective comparison of low level control algorithms that are responsible for positioning the camera.
- Implement a development environment that allows control algorithms to be designed, implemented, tested and evaluated.
- Utilise existing research and develop new control algorithms that can be tested in the experimental environment.
- Study the behaviour of the hardware and the controllers and investigate their interaction in the controlled environment.
- Integrate established high level gaze control algorithms that can interact with real world data and the low level controllers of the experimental active vision platform.

1.4 Outline of the Dissertation

This dissertation is organised as follows:

Chapter 2 reviews fundamental concepts of passive and purposive active vision and then leads into the mechanisms used in active vision, such as foveation, gaze control, stereo and colour vision. The human and artificial active vision systems are then covered in separate sections.

The four key components of the human vision apparatus are introduced and discussed, these are: the eyeball, the protective apparatus, the extra-oculomotor apparatus and the visual pathway. The extra-oculomotor apparatus and the visual pathway are introduced in greater depth, leading into key areas of the dissertation. This covers, in particular, the movement of the eyeball as a result of muscle control and the muscle contraction, stimulated by neural control.

The artificial active vision discussion introduces fundamental mechanical design paradigms, such as: the common-elevation model, the independent gun-turret model, parallel architectures, serial architectures and types of actuation. Thereafter, five active vision systems are introduced and categorised by parallel / serial architecture and monocular / stereo vision. The control of artificial active vision systems is then discussed for established controller approaches. This includes finite state machines, neural networks and control theoretical architectures.

Chapter 3 introduces the design and construction of the new artificial active vision system and a laser targeting system that produces a repeatable target sequence. The design of the vision platform is based on physical properties of the human eyeball and extra-oculomotor system, discussed in chapter 2. The design properties include static balancing, a parallel architecture and opposing linear actuation. The hardware control of the vision system is also based on biologically plausible concepts, where activation potentials cause muscles to contract. The mechanical components are then selected, based on the design requirements. Solenoids provide non-position controlled linear actuation, hydraulic shock absorbers provide linear damping to the solenoids and a gimbal provides a moving platform for the camera. The electrical control is implemented by pulse width modulation (PWM), which allows the simulation of muscle control by activation potentials.

The laser targeting system provides accurately positioned laser targets on a projection screen, controlled by surface reflective mirrors mounted on stepper motor reduction gear drive shafts. The performance of the mechanical systems are tested at the end of the chapter.

Chapter 4 introduces the four software levels that control the active vision system and the simulator of the experimental environment. The implementation level enables the development of gaze controllers, laser target test patterns, low level camera control algorithms and timing configurations for the experiments. The interface level is the interface to the hardware control and the simulator. In this level there is also a parsing facility for the analysis of experimental data. The control level contains the simulator and the programs that interface with the device driver level. The device driver level contains the lowest level software components that communicate directly with the robotic hardware. At the end of the chapter, tests are carried out to validate the performance of the simulator.

Chapter 5 discusses the low level controllers that regulate the positioning of the active vision platform. There are three controller categories: The benchmark controller assesses the performance of laser target controllers. The following non-adaptive and adaptive controllers perform camera control by utilising an error measure, provided by a negative feedback loop. These con-

trollers are introduced in a progressive manner, starting with simple structures and progressing into sophisticated mechanisms that switch between control paths in response to changing target behaviours. The final controllers discussed in this chapter are based on a neural network model of the superior colliculus.

Chapter 6 assesses the performance of the hardware and the controllers discussed in the previous chapter. The hardware is compared to the five vision platforms discussed in chapter 2. The laser target control patterns are then introduced and their performance is assessed by the benchmark controller. All other controllers are evaluated thereafter, using an error graph representation. The convergence of the neural networks is also illustrated by graphically plotting the location of neurons in the network and the associated activation weights. The results of this chapter show that simple controllers can be developed to control vision systems, like the one developed here, that perform saccades, pursuit and even switch between these modes. The hardware also shows performance results that are comparable with similar systems.

Chapter 7 presents the main contributions and limitations of this dissertation. It also suggests possible directions for future work and summarises this dissertation.

Chapter 2

Vision Systems

Vision is a multidisciplinary science and many areas of research are dedicated to understanding, simulating and implementing it. This leads to fundamental questions, such as the ones asked by Aloimonos [3]: “what is vision”, “what could vision be” or “what should vision be”. These are important questions to which each field of research may have its own valid answers. Marr [77] investigated the first two questions in particular, concentrating on the human vision system. He advocated an approach in which a computational theory specifies what expressions exist in sub-processes of human vision, such as stereo and model matching. The human vision system is by no means general [56, 65], but is specifically adapted to certain tasks and the environment it interacts with. The last question: “what should vision be” leads much of the investigation into what type of biological or robot vision is most appropriate for a specific task. The physiological literature covers biological vision systems in great depth and explains many of the components and processes involved in biological vision. The investigation into the operation of the human visual system has attracted particularly wide attention. This is possibly due to the apparent medical benefits that are gained by understanding the system, because of our close relationship and personal interest in it, or because it is a particularly good example of a highly evolved vision system [127]. The literature on robotic vision systems is very diverse and constantly introduces new mechanical designs and increasingly powerful control systems.

This chapter introduces a physiological view of the human eye and contrasts its properties with the technology of available robotic vision systems. This comparison shows some similarities, but also highlights properties of the human eye that are obviously not present in robotic vision platforms. The development of more biologically inspired artificial vision systems may, in the future, help to further the understanding of human eye control and the development of cheaper, faster and more reliable robotic vision platforms.

2.1 Active and Purposive Vision

There are fundamentally two categories of artificial vision systems: *Passive* and *Active Vision*. The physical difference between these two system types can be simply defined as follows:

Passive Vision

> "The typical property of passive vision is that the observer is not capable of choosing how to view the scene, but is instead limited to what is offered, determined by present visual parameters and environmental conditions, including time sampling." [3], p4

Active Vision

> "Active Vision refers ... to strategies for observation. As Aloimonos et al. [4] explain in their seminal paper, it is the observer that is active. Rather than processing snapshots, or sequences of snapshots treated individually, the observer and sensor continually interact. Visual sensory data is analysed purposefully, in order to answer specific queries posed by the observer. The observer constantly adjusts its vantage point in order to allow the sensor to uncover the piece of information that is immediately most pressing." [18], p xv-xvi

Even though the physical differences between passive and active vision are simple, the implications for the way in which images are processed and the way limbs / robotic actuators and eyes / cameras are controlled, is significant. Whereas the passive observer is constrained by the data presented, the active observer can choose what to observe, in order to fulfill a task [31, 36, 64]. This enables the active observer to solve vision problems in a more efficient way than can be done by a passive observer. Aloimonos, Weiss and Bandyopadhyay [4] showed that the problems of: *shape from shading*, *shape from contour*, *shape from texture* and *structure from motion* have many solutions for passive observers and unique solutions for active observers. This assumes that active observers utilise active vision appropriately for the specific vision problems.

So active vision may be able to solve problems that are unsolvable for passive vision, but what is the cost? The control of an active vision system must require additional computational resources. However, if the control of camera systems and image processing were coordinated and it were possible to devise systems and strategies that are capable of selectively focusing the active observer's attention, it may even be possible to reduce the overall vision processing cost, compared to the cost of passive vision. This concept is indeed applied and leads to *purposive vision.*

> "Purposive vision does not consider vision in isolation, but as part of a complex system that interacts in specific ways with the world. It is important to note that if the visual system knows what kind of information is needed and what it will be used for, this permits the system to alter its interaction with the world dynamically in order to make this information more easily available." [3], p5

A property of the human vision system, known as the *fovea*, allows image features to be observed and processed selectively [39, 90]. This small area of maximum resolution vision is moved around the world under control of an elaborate gaze control system, reducing the need to process images that have an evenly distributed resolution. Some areas of active robotic vision also take advantage of foveated image processing, to reduce computational resource requirements [16, 45, 62, 75, 106, 117, 128, 130]. These approaches are varied and range from image transformations to specialised optics and imaging chips, but the selection of where to look does still present problems. Foveal vision is only one imaging mechanism in humans that enables visual attention to be focused and image processing and understanding to be optimised. Other examples,

such as stereo [33, 35, 41, 118] and colour vision [49, 50] have been shown to help image depth interpretation and object identification.

Active vision appears to have certain advantages over passive vision, supported by the evidence of millions of successful biological organisms that use active vision to interact with their environment. However it remains to be determined what the key properties of active vision systems are, since there are multiple ways in which attention can interact in active vision systems.

2.2 Biological Vision

It would be an enormous task to investigate the operational components of all known biological active vision systems, due to their rich diversity and often complex nature. For this reason, the focus is on one vision system that has been examined in depth for many centuries; the human eye. It is complex and still not fully understood, with many unanswered questions remaining to be investigated. This research searches for a wider application of current knowledge about the human vision system, rather than concentrating specifically on these unanswered questions.

According to Kronfeld [59], the human vision system can be broken down into four logical components: The *eyeball*, the *protective apparatus*, the *motor apparatus* and the *visual pathway*. The *eyeball* is three layered, approximately spherical and houses the optical apparatus. This includes the clear front part of the eyeball, know as the *cornea*, the *iris* which controls the aperture, the *crystalline lens* which focuses images and the *retina*, onto which images are projected. The *protective apparatus*, known as the *antechamber* serves two purposes: It protects the eye from mechanical damage and creates an optimal milieu for the *cornea*. The *eyelids* form the outer part of the *antechamber*. The following investigations do not look at these components of the eye in any more detail. It is assumed that the eye produces images and is sufficiently protected against external damage. The *motor apparatus* and *visual pathway*, on the other hand, are of great interest to this research and are discussed in more detail. Their function is covered in the following two subsections.

2.2.1 Motor Apparatus

The healthy human vision system is equipped with a pair of eyes where the anatomy of one eye forms a near mirror image of the other eye. This symmetry allows most properties of eyes to be discussed by investigating one eye in isolation.

The motor system of an eye consists of muscles which individually contain very thin muscle fibers and simple spiral sensors. The muscle fibers show varying electrophysical characteristics and are categorised into *twitch* and *tonic* fibers [89]. The twitch fibers show non-graded, pulsile activity and the tonic fibers show non-pulsile activity with a graded contraction. The spiral sensors, known as *spindles* are richly supplied in the extra-ocular muscles and provide feedback on the degree of muscle contraction [7]. All six muscles of the extra-oculomotor system contain twitch fibres, tonic fibres and spindles. These muscles are attached to the outside of the eyeball and control the rotation position of the eyeball within the eye socket. They are the *medial rectus*, the *lateral rectus*, the *inferior rectus*, the *superior rectus*, the *inferior oblique* and the *superior oblique* muscles. The medial rectus rotates the eye in the horizontal plane nasally and the lateral rectus rotates the eye in the horizontal plane temporally. The inferior rectus primarily depresses

the eye vertically and the superior rectus primarily elevates the eye vertically. The inferior and superior rectus muscles also produce oblique rotation and *torsion*[1]. For a given muscle, the terms *primary* and *secondary* action are used to classify these components of movement, depending on their relative magnitude. The inferior oblique muscle produces extorsion and the superior oblique muscle produces intorsion. Their tonus compensates for the oblique movement and torsion of the inferior and superior rectus muscles. It has been shown [5] that even simple eye movements, such as saccades in the horizontal plane, rely on a controlled interaction of all six muscles.

The extra-ocular muscles can accurately position the eyeball to a small fraction of a degree when stabilising images on the retina, and can be very slow acting when pursuing targets or can move ballistically during saccades. During pursuit movements the eye tends to rotate at the speed of the target, reaching velocity saturation at $25^\circ/s$ to $30^\circ/s$ [100, 131]. Tracking above $30^\circ/s$ becomes increasingly inaccurate and leads to oscillations and saccades [20, 91]. Saccades are high velocity motions of the eyeball and are executed by very fast acting muscle contractions of the extra-ocular muscles. These are in fact the fastest muscles in the human body. According to Fuchs [43] and Robinson [99], the average horizontal 5° saccade takes $30ms$ to complete, with an increase of $2ms$ for every extra 1°. They propose that this linear relationship holds for saccades up to 80°. A range of individuals has been studied and the literature shows a high variation in peak rotational velocities of the eyeball. These lie in the range $350^\circ/s$ to $900^\circ/s$ for healthy eyes [10, 14, 28, 34, 36, 43, 58, 71, 93, 134]. This wide range of observed peak saccadic speeds and results from Becker [14] suggest that the linear approximation is inaccurate. It has been shown that this linear approximation actually exhibits inaccuracies with increasing deviations towards the upper and lower end of the saccadic amplitude range. Other approximations have been formulated to describe these extreme areas more accurately. The contraction speed of extra-ocular muscles does not only deviate between individuals, but also between individual muscles of the extra-oculomotor system. This is especially the case for the inferior and superior rectus muscles. An anatomical reason can be found in the non-symmetric architecture [115] of the oculomotor system, but there are also other influences that cause short term fluctuations in the performance of muscle contractions. Fatigue has a significant influence on performance and can occur after only short periods of increased muscle activation [10, 14, 57, 125]. As few as 30 saccades with a magnitude of 50° are sufficient to cause measurable fatigue. The symptoms are lower saccadic velocity, widely undershooting saccades and longer durations between eye positioning. Fatigue is one factor that may be implicated in the wide range of peak eye velocities that have been measured.

It is remarkable that the arrangement of six muscles is capable of high speeds and precise low speed eyeball positioning. This performance is especially astonishing, considering that the muscle performance is constantly subject to change. The ability to maintain a fairly even eye positioning capability requires a control system that is capable of monitoring and rapidly adjusting to changes of the extra-oculomotor system.

[1] "Rotations around the line of sight (i.e., torsion) are classified as *in*torsion or *ex*torsion as the top of the vertical corneal meridian moves nasalward or temporalward respectively." [5], p33

2.2.2 Visual Pathway and Oculomotor Control

Large regions of the brain are devoted to visual analysis, interpretation and understanding. Much of this information is required to conduct purposeful tasks, such as playing table tennis, driving or reading text and music [25, 66, 67, 68, 69, 108, 111]. As a result of high level processing, the eye can be instructed to perform purposive active tasks, such as foveating on an anticipated location of a ball, or following a fixed point on the road. The eye can also be controlled by reactive processes that are similar to reflexes. The central mechanisms that control eye movements are complex and incompletely understood, but some of the brain regions involved in these tasks have been identified as the *cerebellum*, the *inferior olive*, the *parieto-occipital junction*, the *vestibular system*, the *superior colliculi*, the *lateral geniculate nuclei*, the *motor nuclei* of the extra-ocular muscles and the *reticular formation* [27, 127]. There is no consensus on what information and control is processed in each of these regions, but it is assumed that *cortical regions* of the brain are engaged in eye movement. The *frontal cortex* is apparently related to saccadic movements [99]. The *occipital cortex* is more related to smooth pursuit movements [100]. The investigation of detailed control processes in the brain is very difficult, due to the limitations of techniques currently available. It is possible to detect brain regions which exhibit changes in activity, as a result of some stimulus, but it is currently impossible to determine the exact neural processing that takes place in most of these areas. The brain is too complex, dense and highly connected to allow large regions of neurons to be monitored individually. It is often not necessary to understand the detailed operation and interaction of connected neurons to identify control processes in the brain. In fact, most of the high level control processes are not understood at a neural level, but are often represented as *black boxes* that exhibit specific behaviours. Lower level control processes, on the other hand, are often easier to examine from a neurological point of view, as they are either controlled by fewer neurons or by regions of the brain that exhibit a uniform arrangement of neurons.

Here the investigations only address control processes from a high level and in isolation, without deep neurobiological discussions of brain regions that are believed to cause specific reactions and behaviours. The focus is in particular on processes that are of direct relevance to future investigations within the context of this dissertation. Individual black box behaviours and neural arrangements are addressed which represent widely accepted models within the research community. This also means that the neural pathways and interactions between behaviours and reactions are omitted. The *vergence control* system [101] that enables active depth perception is also not discussed. Processes which influence the control of the extra-oculomotor system and interact with other areas of the body are also not covered. This is, for example, the case with the vestibulo-ocular reflex (VOR) [9, 71, 101], which coordinates head movements with eye fixation, or the *otolith reflex* [113], which generates ocular counterrolling when the head is tilted towards either shoulder.

Parametric Feedback

In the previous subsection it was established that the performance of the extra-oculomotor system can fluctuate under fatigue. A compensatory mechanism, know as *parametric feedback* [7] is in place that measures the muscle extension through *spindles* and feeds signals back to motoneurons that participate in the control of muscle contractions. It is also assumed that the cerebellum [23]

participates in this control mechanism and acts as a sampler between the input from the spindles and the output to the motoneurons. This mechanism provides load compensation [51] and maintains a constant muscle length under varying muscle load. Our mind is not apparently consciously aware of the feedback mechanism [127] and absolute eye position [7]. This *proprioceptive*[2] system not only provides compensatory control, it also helps in providing a linear control relationship between nerve impulses and eye positioning [34, 39].

Eye Positioning Laws

Alpern [6] discusses the definition of *pseudotorsion*, first observed by Donders in 1848 and later defined by Helmholtz as Donders' law. Donders generated a green + afterimage, produced by staring at a red + for a prolonged time. He then looked at a screen in front of him and observed that the + tilted when he looked at top right, top left, bottom left and bottom right positions of the screen. He also found that the degree of tilt was relative to the magnitude of horizontal and vertical gaze. Donders was not able to explain this phenomenon at the time, but Listing, Helmholtz and others developed the necessary equations [113]. It was found that pseudotorsion is not caused by rotation of the eyeball around the line of sight, but by the fact that the horizontal and vertical retina meridians do not coincide with the horizontal and vertical lines in space. Helmholtz defined Donders' law as:

> "Der Raddrehungswinkel jedes Auges ist bei parallelen Blicklinien eine Function nur von dem Erhebungswinkel und dem Seitenwendungswinkel." [38], p158

Translated, this means: "The rotational angle of each eye with parallel lines of sight is a function of only an elevation angle and a rotation angle."

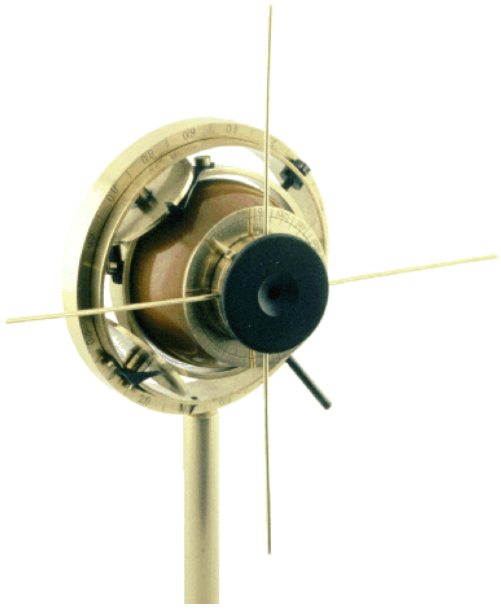


Figure 2.1: Ophthalmotrope developed by Donders in 1870 to illustrate Donders' law. A gimbal suspension is used in this model to make the pseudotorsion visible. The ophthalmotrope is equipped with a camera obscura to obtain images of scenes viewed. This instrument is now exhibited in the museum of the former Royal Netherlands Ophthalmic Hospital in Utrecht.

[2] Sensor stimulation that arises from within tissue.

Listing, in particular, described a mathematical model that assumes two rotational axes which pass through the rotational centre of the eyeball and form an imaginary cross, perpendicular to the line of sight [113]. He also showed that it was possible to start from a fixed position and define two rotational angles that bring the eye into any other valid position [6]. Listing's law holds for vergence control with targets at a constant distance, smooth pursuit and saccadic movements. The law is violated during VOR, otolith reflex and vergence between targets at different ranges [6]. Porrill [92] compared a detailed theoretical model of the extra-ocular muscles with Listing positions and found a reasonable accuracy over a range of $\pm 30^{\circ}$. In 1870 Donders [38] implemented a mechanical model that illustrates Donders' law and, in particular, emphasises the presence of pseudotorsion, figure 2.1. It uses a gimbal suspension to make the pseudotorsion visible. This model does not confirm exactly to the positioning rules, but Listing positions can be reached by a number of transformed angular rotations. [38, 113].

Saccadic Movement

Saccades are believed to be triggered in several regions of the brain, such as the visual, parietal and prefrontal cortex as well as the superior colliculus, the cerebellum and the reticular formation [107]. Saccades can be triggered as voluntary movements, resulting from planned responses or memory cues, or as reactions to an external stimulus, such as visual or auditory influences [36, 44, 93].

From a neurological point of view, the superior colliculus presents a rather interesting configuration which is also well understood [24, 27, 44, 98, 127]. It is a sheet of multilayered neurons that is located in the upper region of the brain stem. The neural arrangement is separated into the upper and lower layers, which are know as the *retinotopic map* and *motor map* respectively.

> "The relation between the location of receptors on the retina which are, *e.g.*, excited by a small light point, and the place of neurons in the upper layer of the superior colliculus that are simultaneously excited, is continuous and topology conserving. This implies that a topographic sensory map from the retina to the ... retinotopic map is realised." [98], p142

In contrast, the lower layer provides sensory locations that are essential for saccadic control. Locations in this layer correspond to saccadic changes in view direction that can be triggered by exciting neurons at the corresponding locations. The excitation can be artificially created by inserting electrodes. Experiments have shown that the direction and magnitude of the saccades invoked are not intensity dependent, but depend on the location of excitation in the motor map [103].

The upper and lower layer of the superior colliculus lie on top of each other, so that local excitations in the retinotopic map are passed to adjacent receptors of the motor map. This means that an excitation in the upper layer, caused by a localised stimulus on the retina, is passed to the lower layer, resulting in oculomotion that leads the fovea to the location of the original retinal stimulation. This neural correspondence led Robinson [103] to formulate a hypothesis, referred to by Burdess and Ritter [24, 98] as the *foveation hypothesis*. This states that the connections between the retinotopic and motor map of the superior colliculus serve to create saccades that centre the object of interest in the fovea. As the physiology of the body changes [44, 53, 73, 80], the control of saccades by the superior colliculus becomes inaccurate and needs to be adjusted. The

cerebellum, apart from many other functions [27, 76], is assumed to be involved in this motor learning and can adapt as the anatomy of the body changes. This controlled learning is influenced by sampling information from the superior colliculus and other regions involved in saccadic control. If a saccade is found to be inaccurate, a *teaching signal* is sent from the inferior olive [44] to the cerebellum, in which weights are adjusted that influence the control of saccades.

Apart from neurological analysis, the saccadic system has been subject to deep control theoretical investigations, and is broadly considered to have four components [14]:

1. A sensor that measures an error between the target position and the eye position in space.
2. A controller that converts the input error into a motor command.
3. A plant which executes the motor command and produces the eye position.
4. A negative feedback loop that reflects the error between the target position and the eye position.

The four listed components and the design of control theoretical models for the saccadic system have evolved over time, in step with increasing knowledge about the control of eye movements. In 1963 Young and Stark [132, 133] proposed a dual model in which the saccadic system was based on *retinotopic feedback*. This and similar models [43, 115] assume that the retina determines the error between the target position and where the eye is pointing, in order to instruct the muscle control system to send signals that minimize this error. Later Robinson [104] formulated the *local feedback* concept of saccadic generation, which rejects the retinotopic scheme and assumes a *target copy* that is used to drive the eyes to the target of interest. This would also allow saccades to be executed in the dark or as a result of auditory stimuli. A number of authors have since elaborated on this model and proposed modifications in line with recent neurological findings, although the basic concept of local feedback remains. Scudder [110] updated Robinson's original model and integrated control components from other parts of the brain that are assumed to be involved in saccadic eye movements. Becker [15] investigated the saccadic control system with respect to the oblique muscle control and derived two plausible control models.

Smooth Pursuit

The control of smooth pursuit is believed to be a fairly recent evolutionary addition to the movement capabilities of the eye. It is widely agreed that the main centre for smooth pursuit tracking resides within the parieto-occipital junction [23], but its origin is still unclear [91]. It is assumed to have originated either from the optokinetic system or from a stabilisation system. The neural control of smooth pursuit also remains widely unsolved. Until 1968 it was widely believed that the smooth pursuit tracking system, similar to the saccadic system, is intermittently sampled. Brodkey and Stark [20] showed that smooth pursuit is in actual fact continuously sampled.

> "The velocity of a target in space is registered by mechanisms in the visual pathway, and relayed to the oculomotor brainstem ..., where the velocity signal is integrated neurally into an eye position signal. The position signal is then combined with the velocity signal, and this combination is sent to the extra-ocular muscles." [91], p139

It is widely agreed that the stimulus for pursuit is directly related to target behaviour and is strongly influenced by target position and target velocity [94, 131]. More recent investigations also suggest that acceleration plays a role in the control of smooth pursuit [91]. Attempts to move eyes smoothly, without such stimuli, result in a series of small saccades. Control theoretical black box controllers have been developed that are suggested to be part of the smooth pursuit system.

Stark and Young's pioneering dual model [132, 133] already mentioned in the context of saccadic eye movement also implements a control path for smooth pursuit. Stark [115] contrasted the pursuit path of his dual model with work by Murthy, Deekshatulu and Foster, who had implemented continuous smooth pursuit controllers. Stark agrees that there is evidence to confirm that the sensory system takes information in continuously and the motor system is capable of producing a continuous output, but he stresses that it is not known whether the neurological controller samples intermittently or continuously. Fuchs [43] combined the dual model with the pursuit controller by Foster. His representation shows the redundant, sampled control structure and highlights the revised continuous addition. Other models [23, 95, 131] also contain continuous control paths, but apart from models that view smooth pursuit as one entity, other approaches [26, 91, 102, 126] explore individual behaviours within the smooth pursuit system. In particular, these newer control approaches take into consideration not only target velocity, but also target position.

2.3 Machine Vision

Early attempts of machine vision aimed at developing systems for general purpose vision. The approach was to extract as much information as possible out of one or a number of scenes, captured by cameras mounted in fixed locations [114]. Although this research resulted in the development of interesting new algorithms, the systems were not robust (see section 2.1).

Over the last fifteen years, machine vision has witnessed a remarkable shift from passive to active systems, with the development of new algorithms, new image acquisition systems and a tremendous increase in computer power [3, 4, 81]. This transformation saw a decline in bulky fixed systems and the emergence of smaller systems and autonomous agents that are able to use vision as a means to interact with the environment. Similar to biological agents, these artificial agents are able to move around their environment, perceive objects, instruct actuators to manipulate objects and develop maps that help them to plan actions and return to visited locations.

Even though active machine vision is a fairly recent science, the advances are remarkable. The following two subsections look at some of the capabilities and advances in this field. The focus is, in particular, on components that are directly involved in active vision, such as camera systems, image representation and the algorithms that control the visual interaction process. Manipulators and platforms that enable agents to move through and change the environment are not covered here.

2.3.1 Active Vision Platforms

The hardware design of artificial active vision systems is very diverse and utilises a range of technologies, depending on the task the platforms have to fulfil or the capabilities that are to be investigated. A range of design concepts and hardware architectures are introduced here that can be found in many systems currently available.

In the following context, the terms *tilt*, *vergence* and *pan* are used to refer to movements that can be performed by the vision systems. The platforms described here have unambiguous tilt axes and it is sufficient to use the term tilt to refer to vertical movements of the platforms. The horizontal movement of cameras here is platform dependent. The term vergence is used to describe the horizontal movement of individual cameras that are part of a stereo vision system. The term pan is used to refer to the horizontal movement of stereo vision heads that have cameras, which are individually capable of vergence movements. Pan is also used to refer to monocular cameras that have only one rotational axis that enables horizontal movements.

One of the simplest active vision systems consists of a camera that is capable of pan and tilt movements, similar to many of the police surveillance cameras that can be seen in public areas. Such monocular camera platforms form the basis for stereo vision systems, and it is an obvious progression to mount two monocular cameras next to each other in order to build a stereo system. One distinguishes between two types of stereo head configurations. The first model, shown in figure 2.2a is known as the *common-elevation* configuration where both cameras have individual vergence capabilities and a common tilt platform. The second model, shown in figure 2.2b is known as the *independent gun-turret* configuration where both cameras have independent vergence and tilt capabilities. Murray [84, 112] suggests that there are no extra benefits to machine vision in using one architecture over the other one.

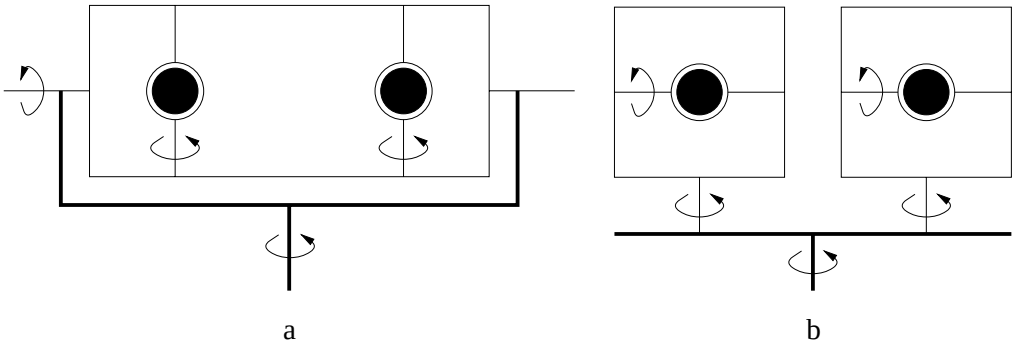


Figure 2.2: Two common stereo head architectures. (a) the common-elevation model and (b) the independent gun-turret model. (Redrawn from Murray.)

The movement of cameras, on monocular or stereo platforms, is generated by actuators that apply a force between two or more kinematic links. A widely used configuration known as a *serial architecture* has proven to be very successful, due to the straightforward design, simple kinematic models and reduced problems with singularities. Such serial architectures are categorised by a chain of actuators that act upon kinematic links [47]. An inherent problem with this architecture is that the weight of actuators and links higher up the kinematic chain are moved and carried by actuators lower down the chain. This leads to potentially less stiff structures and can require larger actuators at the bottom of the chain. Serial systems are not the only way to build vision systems, in fact biological vision systems, such as the human eye uses a *parallel architecture*. Such systems are widely characterised by several actuators connecting from a base to an end platform in parallel. This makes the systems more rigid, more compact and allows a more accurate and faster performance. This architecture is not widely used in artificial active vision systems as the mechanics

are often complicated to design, have complex singularities and often difficult kinematic models [21, 47, 48]. Parallel architectures do also not scale up well for the implementation of stereo vision heads. It would be very hard to design a parallel architecture stereo vision head that was capable of tilt, vergence and pan movements, although a parallel architecture head with tilt and vergence capabilities does exist [116].

> "The penalty of having a parallel architecture is that it makes the device more complex. Indeed, adding a fourth degree of freedom, a global pan (neck) joint, and still keeping to the parallel drive architecture would be challenging." [116], p6

A combination of serial and parallel architectures is more feasible and has already been demonstrated in nature, i.e. the human eyes and head movement. Artificial active vision stereo heads are still widely based on serial architectures [16, 29, 60, 61, 74, 112].

The actuators used in parallel or serial architectures, such as stepper motors, direct current (DC) and direct drive motors, usually have a defined positioning resolution. They apply movements to the kinematic links of the active platforms, either through direct connections, gears, belts or other means of passing on activation forces. The fixed resolution allows cameras to be repeatably positioned in defined locations, without requiring any sensory feedback. Camera positions in space can be derived through the kinematic model, which presents a mathematical representation of the camera platform kinematics.

The performance of active vision systems varies considerably from one platform to another, depending on the intended application, resulting architectural design, implementation and control strategies. The following table lists the mechanical performance characteristics of five different

	Parallel Architectures			Serial Architectures	
	Monocular Vision		Stereo Vision		
	HPV [21]	Agile Eye [47]	CeDAR [116]	ESCHeR [61]	Yorick [112]
Max Pan Acc	$72000^\circ/s^2$	$20000^\circ/s^2$	$20000^\circ/s^2$	$4000^\circ/s^2$	$3000^\circ/s^2$
Max Pan Vel	$600^\circ/s$	$1000^\circ/s$	$800^\circ/s$	$140^\circ/s$	$150^\circ/s$
Max Pan Ang	120°	$140^\circ \pm 30^\circ$	90°	200°	360°
Pan Res	0.01°	$0.135^\circ \pm 0.045^\circ$	0.01°	0.0044°	0.00018°
Max Tilt Acc	$72000^\circ/s^2$	$20000^\circ/s^2$	$18000^\circ/s^2$	$14000^\circ/s^2$	$5000^\circ/s^2$
Max Tilt Vel	$600^\circ/s$	$1000^\circ/s$	$600^\circ/s$	$350^\circ/s$	$400^\circ/s$
Max Tilt Ang	180°	$140^\circ \pm 30^\circ$	90°	90°	360°
Tilt Res	0.01°	$0.135^\circ \pm 0.045^\circ$	0.01°	0.0145°	0.00036°
Max Verg Acc	n/a	n/a	n/a	$16000^\circ/s^2$	$6000^\circ/s^2$
Max Verg Vel	n/a	n/a	n/a	$400^\circ/s$	$400^\circ/s$
Max Verg Ang	n/a	n/a	n/a	100°	360°
Verg Res	n/a	n/a	n/a	0.0125°	0.00036°

Table 2.1: A comparison of hardware performances characteristics for five different active vision platforms. This table contains a selection of architectures that can be found in most active vision systems currently available.

active vision systems. The platforms were selected on the basis of design diversity and the availability of relevant research material. These platforms are categorised into serial / parallel architectures and monocular / stereo vision systems. The first vision system, HPV, is defined as a

high-performance camera platform [21] and is driven by two actuators. It has a simple kinematic model in which pan movements are controlled by one actuator and tilt movements are controlled by synchronised control of both actuators. The main features of this monocular, parallel architecture vision platform are very high acceleration and a low mechanical error, due to the use of backlash free drive belts. Agile Eye [47] is another parallel architecture, monocular camera system, but with significantly different properties. This system is driven by three motors that enable fast acceleration and high velocity. The platform has a complex kinematic model and pan and tilt movements are controlled by a coordinated activation of all three motors. In this model, the centre of camera rotation is not fixed and is moved in a working envelope that has the shape of a cone. CeDAR [116] is a parallel architecture stereo vision head with three actuators and is based on the common-elevation model. This vision system was inspired by the previous two platforms. The benefits of the parallel architecture are reflected in the high velocity and acceleration, which are within range of the previous two monocular vision systems. The use of drive belts also reduces the positioning error and the kinematic model is similar to that of the first vision system. Two actuators drive the vergence movements of the cameras and one actuator drives the movement of the common tilt platform. ESCHeR [61] is a serial architecture, stereo vision head with three DC motors and is based on the common-elevation model. The acceleration and velocity of this platform are lower than those of the parallel architecture platforms. The kinematic model of this platform is relatively simple, as two motors control the vergence movements of the cameras and one motor controls the common tilt movement. The special design characteristics of this platform are foveated wide angle lenses. The final platform, Yorick [112] has a similar kinematic model and drive system to that of ESCHeR but does not have foveated wide angle lenses. Yorick is designed to have a larger rotational range around all axes. Tilt, pan and vergence movements can be performed in full 360°.

Apart from the diverse designs and implementations of active vision mechanics, there are also major differences in camera designs. These range from cameras with standard charge coupled device (CCD) arrays to cameras with foveated CCD chips [88]; and from camera optics with fixed lenses and set focal lengths, over fully motorised lenses to foveated lenses [16]. The design of mechanical hardware platforms is heavily influenced by the type of camera used. Smaller and lighter cameras allow for smaller and faster camera platform designs.

2.3.2 Active Vision Control

The control of active vision systems today, is implemented by a combination of hardware and software components. The hardware usually consists of a computation device on which software instructions are executed, an image capture component and an amplification unit that converts low current control signals into high current actuator drive signals. The design of hardware control systems is time consuming, potentially complex and not within the scope of this review. In the following context, hardware devices are therefore considered to be black box entities. It is assumed that the hardware is capable of executing software programs, capturing images and driving active vision platforms. The software components are of greater interest. A significant amount of research has focused on the design and implementation of new software algorithms for the exploration of active vision capabilities. Software also has a short turn around time between im-

plementation and evaluation.

There is no defined right or wrong way by which active vision systems can be controlled, but many software architectures are emerging that use two or three heavily interlinked tasks [19, 22, 30, 78, 85, 86, 87]: *visual processing*, *gaze control* and *actuator control*. Visual processing and gaze control or gaze control and actuator control are sometimes so heavily interlinked that they form a single control unit, even though they perform distinctly different control tasks. Visual processing is involved in the analysis of images, captured by the vision hardware. This often involves the detection of moving features between frames and the extraction of predominant image characteristics. Gaze control uses this information to select image regions of interest. The selection process is task specific and depends on the objectives that are to be met by the system. If it is identified that the direction of gaze is to be altered, the actuator control is instructed to move active vision components accordingly. This process normally utilises a kinematic model, which from an engineering point of view, is well understood [52]. Hence most of the artificial active vision literature does not provide much information on this task. Visual processing and gaze control, on the other hand, are still very much exploratory fields and new control architectures are constantly emerging. Visual processing historically originated with passive vision and many of the algorithms developed then are now used in active vision [46, 52]. Gaze control emerged with the development of active vision systems and can be considered to be the most recent control component within active vision systems. The main interest here also focuses on gaze control.

Artificial gaze control has been heavily influenced by behaviours present in biological vision systems. Smooth pursuit and saccadic control, in particular, are a feature of most gaze control systems. Vergence control can also be found in most stereo vision systems. The implementation of the underlying gaze controllers is mostly inspired by observed eye movements, but physiologically plausible black box controllers and understood neural configurations are also implemented. The control models by Robinson [101], for example, have been implemented on the Harvard head [31, 32]. The performance of the controllers was then tested with blob tracking experiments. The results showed that the system operated as desired, although it was not specified in detail what the desired results were supposed to be. Vergence and saccadic modes resulted in the fixation of blob features, as they moved around in space.

Neurologically plausible gaze controllers, such as that of the superior colliculus have been implemented in simulation [24, 98], but only recently has it been attempted to test this type of controller on an active head [12]. These controllers simulate the direct control of motor-neurons as a result of visual stimulation. The major difficulty in implementing such control models lies in the fact that the output signals need to be converted into a representation that can be used to control available vision platforms.

Other gaze control architectures use a range of engineering and artificial intelligence techniques. The active vision platform Yorick [85] has been used to explore the use of finite state machines (FSM). Bradshaw [19] utilised the states: *inactive*, *saccade-wait*, *saccade-active* and *pursuit-active* to control smooth pursuit and saccadic movements. The states were entered depending on the information received from visual processing, with the gaze model set to one of four modes: *saccade-only*, *pursuit-only*, *test* and *off*. This control model allowed the investigation of smooth pursuit or saccadic camera movements. Murray [83] developed a more advanced

gaze control system built on these concepts. This model utilised six states: *inactive*, *wait*, *saccade*, *pursuit*, *panic*, *reset forward* and was used to test the transition between smooth pursuit and saccadic movements. The system could be run in one of four modes: *saccade-only*, *panic-only*, *pursuit-only*, and *test-saccade-to-pursuit*.

Christensen, Horstmann and Rasmussen [30] developed a biologically plausible, control theoretical approach to active vision, for the AUC camera head [29]. This model was organized in a hierarchical structure with three dedicated control processes: *fixation*, *pursuit* and *attention*. Fixation formed part of the lowest control level and was designed to detect image level features. This used both monocular and binocular techniques. In monocular vision the image focus plane was adjusted to intersect with the object of interest. In binocular vision, fixation was achieved by image disparity and vergence control that focused on the object of interest. Pursuit formed part of the higher level control tasks and also utilised a combination of monocular and binocular vision. For the case of monocular vision, retinal slip provided an error measure of how far image features had moved. The binocular vision system used an initial image match to obtain depth information and then tracked objects within scene coordinates. The error measure was given by the distance between fixation points and the object location. Attention was also specified as a higher level process and the selection of targets was a response to interpretations of the scene. This mechanism was driven by spatial and temporal information. The integration of these control processes was implemented by an inner control loop in which fixation formed the fundamental operational component. The higher level control operations, pursuit and attention, formed outer loops and were linked into the inner loop. The different loops in the system were coupled by switching between them or using summation nodes. Results showed that switching caused discontinuous "jerky" motion, whereas summation resulted in more smooth motion.

These few operational control concepts are a small example of how diverse active vision head control strategies can be, and it is to be expected that many more algorithms will emerge in the future.

2.4 Discussion

Artificial active vision builds on a wide range of knowledge from centuries of research. Around 1500 Leonardo da Vinci drew sections of a man's head, showing the anatomy of the eye[3]. In 1780 Galvani experimented with electricity to make frog muscles contract [119]. In 1848 Donders' observations set the foundation for Listing's law that describes the positioning behaviour of the human eyeball and many subsequent modern vision systems. In 1945 von Neumann demonstrated that a computer could have a simple, fixed physical structure and yet be able to execute any kind of computation effectively by means of programmed control [120]. In 1975 Garland and Melen attached a camera to a computer to develop a security system [121]. This list of historical contributions could be extended significantly, but it is intended to just convey the diversity of disciplines from which current artificial active vision research can draw. It cannot be emphasised enough how complex and powerful active vision is. It has reduced computational requirements for image processing, compared to passive vision, but still has significant capabilities for observing and rapidly

[3]The red chalk, pen and ink drawing ($20.2cm \times 14.8cm$) is now exhibited in the Royal Library of Windsor.

evaluating the environment.

> "Unfortunately, it is all too easy to overlook the exquisite mechanism and control of the human eye - on the mechanical side it has low inertia and can rely on muscles for its actuation, muscles which have properties unmatched by commercially available motors, particularly with regard to low friction, high efficiency, high power / size ratio, co-actuation to control stiffness, and rapid acceleration. On the control side, the physiological behaviours themselves are not fully understood, are probably non-linear, and are therefore difficult to simulate." [84], p155

In nature, systems have evolved to be well adapted to specific environments and tasks. Similar paradigms apply to artificial active vision. Many vision platforms are developed to explore the capabilities of active vision and use biology merely as an inspiration. Much of this work is based on observed behaviours of biological systems but it is not significantly concerned with the underlying concepts that govern the behaviours. Other active vision work is more theoretical and looks at nature to develop biologically plausible models of active vision. Yet other research uses the theoretical results and applies them to engineering and robots to prove or disprove them. This research can even help further the understanding of biological vision systems.

> "Using robots as physical models to test biological hypotheses has particular advantages over computer simulations. The effects of a real environment on real sensors are reproduced directly, and thus mechanisms that depend on this interaction can be more rigorously evaluated. It is often found that the internal processing requirements are substantially simplified by having the right physical relationship with the task environment. Thus results from robots can contribute to biological understanding." [124], p2

Many biologically plausible approaches to artificial active vision focus on the design of visual processing and gaze control but often lack sufficiently strong biologically inspired mechanical designs. The development of mechanical hardware is widely implemented with strong structured and calculated engineering techniques. There is no doubt that this approach is valid and highly successful, but does this approach really do biological vision justice?

> "If there is a dirty, cheap trick, use it, that is how biology does it." Quoted from Chris Malcolm, in a lecture on Intelligent Assembly Systems, DAI, University of Edinburgh, 14 February 1997.

Biology is full of inaccuracies and errors, which in actual fact are a fundamental part of successful evolution. Nature continuously introduces small changes into systems and selects those that are fittest in a certain environment. These are then used to develop future systems with similar, but slightly modified properties. By doing so, evolution is capable of developing surprisingly elegant and efficient solutions for seemingly difficult tasks. These solutions can be very complex and even consist of finely tuned interactions with other entities in the environment. Although evolution is very powerful there are high costs. Biological beings with sophisticated vision systems have evolved on this planet over millions of years. During this time, vast numbers of prototypes have been tested and discarded.

Most engineering disciplines cannot take advantage of such evolutionary development concepts, due to time restrictions and the wasteful selection strategies. Engineered systems are usually

individually designed and assembled from a range of manufactured components. This often involves the interaction of a range of engineering disciplines which individually produce subsystems that are integrated into a complete system. This can lead to expensive, overengineered products that are expected to work to defined design specifications. This approach has produced very good vision systems [8, 21, 29, 47, 48, 74, 112, 116]. Accurately controllable actuators allow repetitive movements without the need for feedback, with accelerations and velocities comparable to those found in biological systems. Artificial active vision platforms are also not designed to fatigue or significantly change their physical properties over time. These design constraints have little in common with the properties of the human extra-oculomotor system and most other biological systems. Biological muscles do not reproduce accurate repetitive eyeball positioning without feedback. Muscles fatigue, and the physiology of the extra-oculomotor system does change over time.

It is now possible to step back from the conventional engineering approach to active vision and deliberate about what properties of the extra-oculomotor system could have a stronger influence on the design of new vision systems. It may be possible to utilise solutions that have emerged through biological evolution to build elegant new systems. Overall, it is questionable if the current design philosophy of active vision systems is really necessary for the development of active vision platforms. There appears to be a possibility of approaching active vision from a more minimalist direction, utilising components that are possibly cheaper, smaller and less reliable. For example, why should an artificial active vision system not fatigue? It may be necessary for the vision system to adjust and learn as platform behaviours change, and biology has given many examples that do this very well.

It is the objective of this research to investigate the possibilities of developing an active vision platform that is very much inspired by the dynamics and performance of the extra-oculomotor system, reproducing behaviours that have so far only been tested in simulation. This essentially involves developing adaptive control mechanisms that interact with the hardware and are independent to existing gaze control algorithms.

2.5 Summary

This chapter introduces vision as a multidisciplinary science and shows some of the fundamental questions that are asked in this field of research. A distinction is made between passive and active vision, indicating some scientific approaches made. The computational and interaction benefits of active vision are also highlighted. This leads to an investigation into the current understanding of biological active vision with an emphasis on the human oculomotor apparatus, visual pathway and oculomotor control. The investigation then focuses on artificial active vision and introduces a number of examples that show the current progress in the field. Some control architectures that underlie the control of these platforms are introduced. It is suggested that biological vision has a significant influence on the development of active computer vision. The discussion draws on a comparison between biological and active vision, emphasizing that each field could benefit significantly from the other. It is also shown that advances in artificial active vision are very much influenced by established mechanical engineering concepts, rather than being directly influenced by the physical properties of the extra-oculomotor system. Questions are posed that show limita-

tions of current engineering approaches to biologically plausible active vision platforms.

The following chapter introduces a mechanical and electrical hardware concept that exhibits more biologically plausible properties than are found in conventional active vision platforms.

Chapter 3

Physical Environment

Chapter 2 highlighted the need for an artificial active vision system that would allow the observation of biologically plausible smooth pursuit and saccadic eye movement control strategies. This chapter introduces such an artificial vision system and focuses particularly on biological design plausibility and practical means by which these design objectives can be met. Manufacturing details are not covered here, but further information is provided in appendices B, C and D.

To achieve an objective assessment of the active vision system and its controllers, it is necessary to implement an experimental environment [109], in which accurate and repeatable image sequences can be generated. This environment and the active vision system allow a range of eye control concepts to be tested, which before could only be tested in simulation.

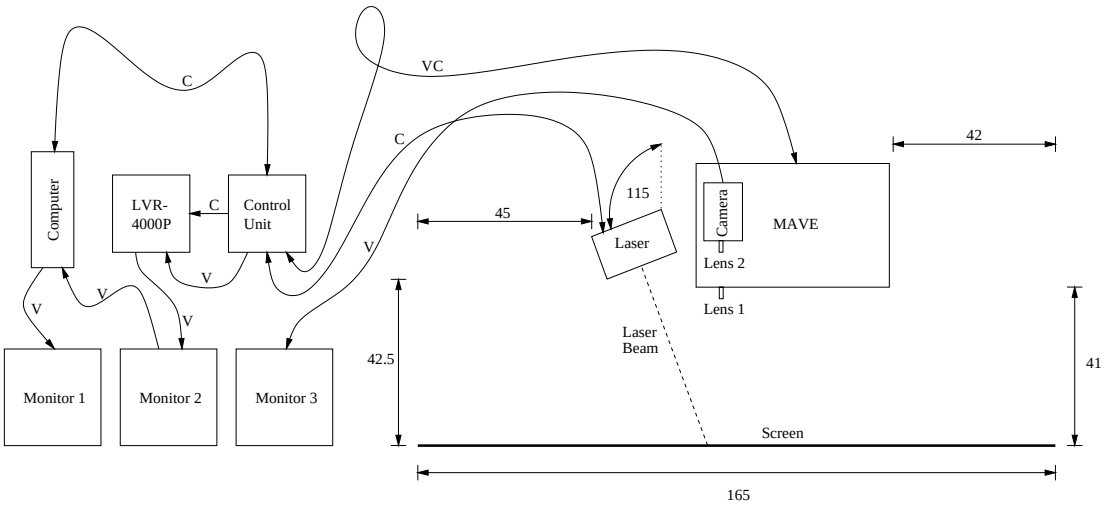


Figure 3.1: Experimental set up (Unit size: cm). Connections marked with a *C* are control communication links. Connections marked with a *V* are video or SVGA communication links. The arrow heads mark the direction of communication.

Figure 3.1 shows a diagrammatic representation of the experimental environment developed here. This overview contains much detail and should be viewed as a reference point, that will clarify the purpose and location of individual hardware components, which are discussed in the

following sections.

The layout of the experimental environment is structured in two parts: the monitoring and control area on the left and the experimental or testing area on the right. The control area consists of three *Monitors*, a *Computer*, a laser videodisc recorder (*LVR-4000P*) and the *Control Unit*. *Monitor 1* provides the visual display to the *Computer*. *Monitor 2* presents the image that is observed by the mechanical Monocular Active Vision Eye (*MAVE*) (*Lens 1*). *Monitor 3* shows a fixed wide angle view of the experimental environment (*Lens 2*). The *Computer* controls the operations and experiments in the testing area, by interfacing with the *Control Unit*. The *LVR-4000P* records high speed image sequences that are captured by the mechanical MAVE. The testing area consist of a *Laser*, a *Screen*, the *MAVE* and a fixed *Camera*. The *Laser* produces moving targets on the *Screen*. The *MAVE* performs smooth pursuit or saccadic movements as a result of visual stimulation through laser targets, projected onto the *Screen*. The fixed *Camera* provides a safe overview of the experimental area, when the *Laser* is switched on [1, 2]. The *LVR-4000P* is triggered by the *Control Unit* and records image sequences at a rate of 50 fields (25 frames) per second.

3.1 Mechanical Design of the Monocular Active Vision Eye

The mechanical Monocular Active Vision Eye (MAVE) fundamentally consists of a charge coupled device (CCD) camera that is mounted on a moving platform, which is positioned by actuators. This concept is common to most artificial active vision systems, but the detailed designs vary considerably from one platform to the other. To provide a biologically realistic design and a justification for a new approach it is first necessary to establish how well current active vision systems compare to the extra-ocular system. This comparison should provide relevant information about the biological plausibility of existing designs and help define key properties of the human eye in a set of engineering terms. These terms can then be used as fundamental properties that may be included in the design of the MAVE.

The following six points contrast abstracted human extra-ocular features with properties of commercial active vision platforms. This is a simplified, top level comparison. A detailed comparison would raise many issues that could not be covered in this context:

1. The fundamental actuator architecture of the human extra-oculomotor system consists of six muscles, which apply their force in parallel to one platform (eyeball). In engineering terms, this is referred to as a *parallel architecture* [47, 48]. Most mechanical camera platforms use a *serial architecture*. This means that one actuator connects to one platform which then connects to another actuator and platform and so on. The difference is illustrated in figure 3.2 [8, 47, 48].

2. The operation of mechanical camera platforms, whether serial or parallel, is inherently *position controlled*. This means that it is possible to point the camera precisely at a defined location, without requiring proprioceptive or camera image feedback. The human vision system uses visual *feedback* from processed images and muscle contraction measurements from muscle spindles to help trigger controlled eyeball positioning [37].

3. The actuators used for mechanical camera platforms normally control two directions of motion, usually clockwise and anticlockwise rotation. Extra-ocular muscle contraction is only applied in one direction. The human eye uses six muscles to control all three degrees of rotational eyeball freedom. This type of configuration is referred to as *opposing actuation*.

4. In most camera platforms, the weight of the camera is carried directly by the actuators. This means that the actuators must continuously apply a force, or the actuators are constantly under a force applied directly by the weight of the camera. The weight of the human eyeball is carried by a concise cavity in the skull. The muscles only apply a force to change the direction in which the eye is pointing. This type of configuration is referred to as *statically balanced*[1].

5. Commercial camera platforms are available with varying degrees of freedom. Figure 3.2 shows two such platforms. They are monocular active vision heads, one with two degrees of freedom, the other with three degrees of freedom. The extra-oculomotor system can control *three degrees of freedom* that allow the human eye to pan, tilt and roll, although according to Listing's law [92], it is possible to define the position of the human eyeball with only two independent entities (angles of rotation).

6. The control signals, sent to the actuators of camera platforms, can vary from multi-strand continuous direct currents (DC) to *pulse width modulated (PWM)* and alternating currents (AC). It can be argued that *PWM* control signals have the closest resemblance to activation potentials that travel along axons to muscles[2].

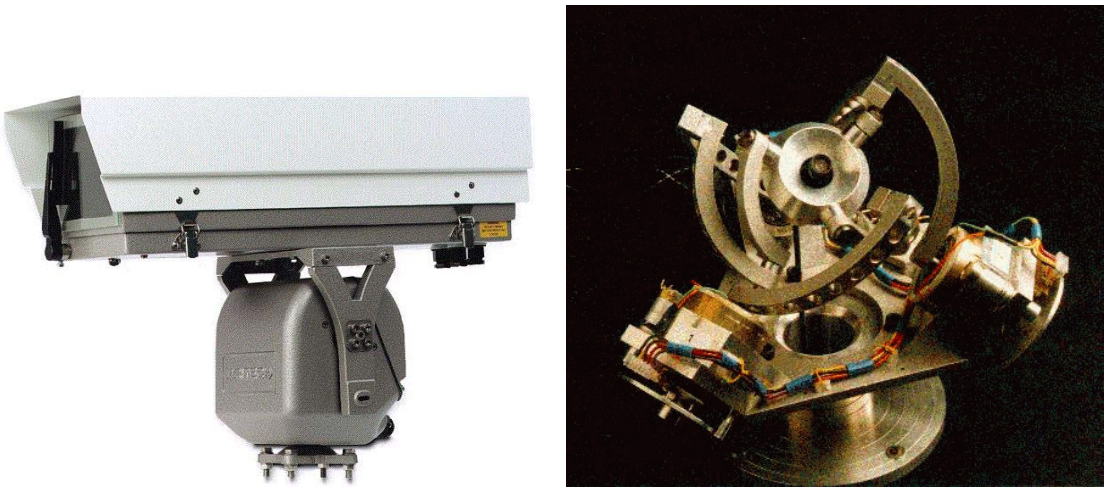


Figure 3.2: The left image shows a monocular active vision head with a *serial architecture*. The two actuators for the pan and tilt movement are located between the pan and tilt joints. The right image shows an active vision head with a *parallel architecture*. Three actuators are connected directly between the base and the camera.

Figure 3.2 shows two typical mechanical camera platforms that are used for active vision. Both systems are *position controlled* and have *bidirectional actuators*. The platform on the left has a *serial architecture*, is driven by geared AC or DC motors and can pan and tilt. The platform on the right has a *parallel architecture*, is driven by three stepper motors and can pan, tilt and roll. These and other commercial platforms [8, 16, 29, 42, 74, 85] lack many of the high level

[1] "Static balancing consists in ensuring that the actuators do not contribute to supporting the weight of the moving links for any configuration" [47], p.7

[2] It is important to note that activation potentials are low current signals which cause biochemical processes to generate energy for muscle contraction. *PWM* on the other hand are pulsed, high current DC signals that directly drive physical actuators.

biologically plausible properties that were identified in the six comparison points. This certainly gives rise to the fact that alternative designs could be considered, which may include more of the biologically plausible features. One could assume that it is possible to implement all six identified features in one platform, but the development here is restricted by time and a fixed budget. So to what extent do the limited resources allow the implementation of an experimental mechanical *Monocular Active Vision Eye* that could be claimed to have similar properties to those of a human eye? Ideally it should be similar enough to test control algorithms that are proposed to underlie smooth pursuit and saccadic eye movements in humans. As six favorable eye properties have been established, that could be utilised in the design of an artificial eye, the next step should investigate how such properties could be combined. The approach taken here looks at the anatomy of the extra-ocular system as a basis around which to design the robotic platform. Functional mechanical components are then discussed that may be able to replicate the behaviours of their biological counterparts. Figure 3.3 [40], p217 shows an anatomical view of the human eye.

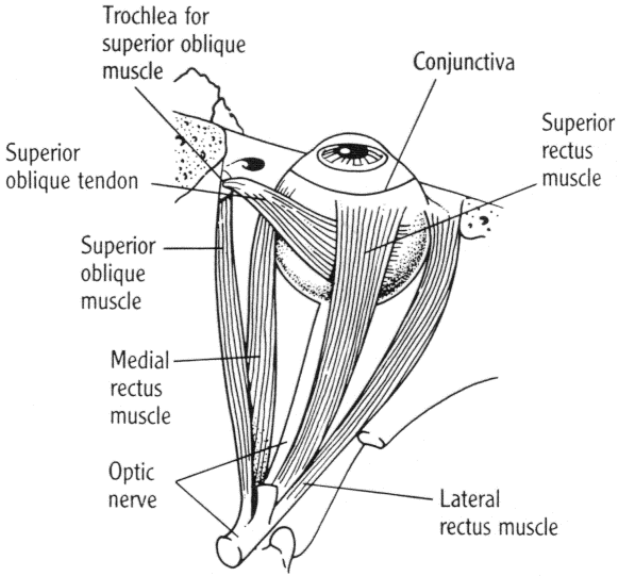


Figure 3.3: An anatomical top view of the human, right extra-oculomotor system.

The eyeball sits in a concise cavity within the skull, which is bedded with fatty tissue. This allows the eyeball to rotate freely. The eyeball and the surrounding tissue are also heavily over-damped [99], which insures that the performance of the eyeball is little affected by the mass itself. The ball in socket mounting concept also means that the eyeball is rotationally holonomic[3] with three degrees of freedom. However, under most circumstances, the eye only performs pan and tilt movements. For design simplicity, it is assumed that a positioning platform with two degrees of freedom, namely pan and tilt is sufficient. An engineering solution, know as a *gimbal*, implements a moving base with two degrees of freedom, which allows pan and tilt movements. The gimbal is also one of the earliest mechanical models, figure 2.1, that was used to explain the behaviours and laws of human eye movement.

[3] An active entity is said to be holonomic if there are at least as many controllable degrees of freedom as possible degrees of freedom.

The human eye uses the medial and lateral rectus muscles to control horizontal movement, and the superior and inferior rectus muscles to primarily control vertical movement. Due to the physical alignment of the superior and inferior rectus muscles, they also generate an oblique component of movement, which is partially compensated for by the superior and inferior oblique muscles [37]. As the gimbal only allows two degrees of freedom, namely pan and tilt, there is no requirement for the control of an oblique positioning component. The mechanical MAVE therefore only needs actuators that control vertical and horizontal movement.

The extra-ocular muscles generate linear motion and have a response time of $\sim 125ms$ [100]. Ideally, the actuators in the mechanical MAVE should generate linear motion and react at a similar speed, allowing the mechanical MAVE to exhibit response times close to those found in the human eye. Solenoids can apply linear motion and have rapid response times. These actuators are not position controlled and normally move a plunger between two physical stop positions. Their use in the control of smooth pursuit and saccadic eye movement has not been investigated before. However, their mechanical properties, in conjunction with mechanical damping, appear to be similar to those of biological muscles [11]:

- Solenoids and muscles actively contract under application of electric pulses / activation potentials.
- Solenoids and muscles can be passively compressed or expanded freely, when no electric pulses / activation potentials are applied.
- Solenoids and muscles cannot expand themselves and must be expanded by an opposing force[4].
- Solenoids and muscles can convert electric pulses / activation potentials directly into physical movement[5].

Even though the properties of solenoids are not examined for the task of saccade and pursuit they are to fulfil here, they are simple and appear to have biologically plausible features.

The extra-ocular muscles are constantly under tension and are attached directly to the eyeball by short sinews. This prevents the muscles from building up slack. In the mechanical MAVE design, it is possible to use rods to pass on actuation from the solenoids to the gimbal. This simplifies the future design of control algorithms, as it is not necessary to implement a mechanism for taking up slack from some flexible connection mechanism. However, this also complicates the connection between the rods and the gimbal, as it is necessary to find a connection system with three degrees of freedom and a small error tolerance. Rod-end bearings provide the degrees of freedom required and have a negligible error tolerance. They are also resilient to the force that can be applied by the solenoids.

The components selected so far do not provide a facility for damping. A gimbal is usually mounted on bearings to prevent friction, and the solenoid plunger moves freely in and out of a coil. This means that extra components have to be selected for damping. As mentioned above, the human eye is heavily over damped. It would therefore be within our design paradigm if

[4]Some solenoids can be fitted with permanent magnets or springs to allow autonomous expansion.

[5]It should be noted that the energy for muscle contraction is provided by adenosine-diphosphate (ADP) and adenosine-triphosphate (ATP) and not by electricity, as is the case for the solenoids.

damping could be applied directly to the gimbal, as the damping of the eye is applied directly to the eyeball. Miniature shock absorbers, used in model racing cars can be adjusted to different levels of damping. They also operate linearly, which should make the integration with the linear actuators relatively easy.

With the availability of core components that allow the simulation of biologically plausible behaviours, it is apparent that there is a good chance for developing a mechanical MAVE that implements most of the features identified in the list of six comparison points.

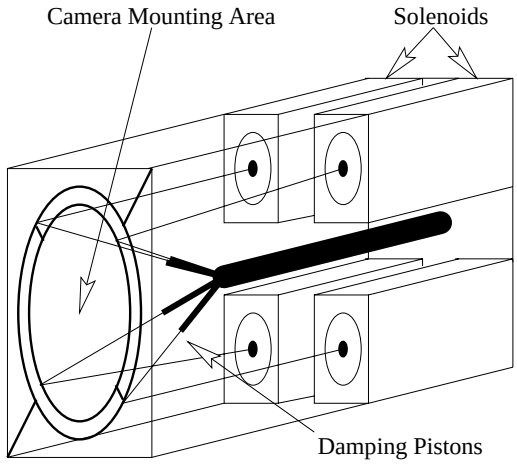


Figure 3.4: Initial schematic design of the mechanical Monocular Active Vision Eye (MAVE). This design consists of a gimbal with the camera mounting area, the damping pistons and the solenoids that generate camera movement.

Figure 3.4 shows the first diagrammatic design of the mechanical MAVE, based on the components selected so far. The *Camera Mounting Area* is, in effect, the inner ring of the gimbal. The four *Solenoids* represent muscles and the four rods extending from the *Solenoids* to the gimbal represent sinews. The *Damping Pistons* also connect to the gimbal and implement the mechanical damping. Apart from the reduced degrees of freedom, from three to two and the reduced number of actuators, from six to four, this design incorporates all derived features, even *static balancing*. *Static balancing* is implemented by counterbalancing the weight of the rods and their fixtures with the weight of the CCD camera.

All components, except for the gimbal, are standard components and are produced in bulk. Purchasing standard components can help reduce cost. A gimbal, even though widely used, is not available in the required dimensions and has to be designed and manufactured separately. A wireframe model of a gimbal was constructed for this purpose. During the manufacturing phase, this model became a useful tool for illustrating key requirements. Appendix C illustrates gimbal components and their assembly.

The long lead time for some components meant that the gimbal was completed before all other parts had arrived. In an extreme case, this meant that the manufactured dimensions of the gimbal were not adequate for the integration, in accordance with the initial design. The area behind the gimbal was too restricted to mount the shock absorbers and the rods that connect to the solenoids. A design decision was made, that would alter the initial overall design. The linear operation of

the solenoid pistons was extended through both ends of the coils, allowing the damping pistons to be moved from behind the gimbal to the other end of the solenoids. Figure 3.5 shows this new design. In effect, this approach provides a much more modular structure than was initially conceived, allowing all mechanical components to be more accessible and maintainable.

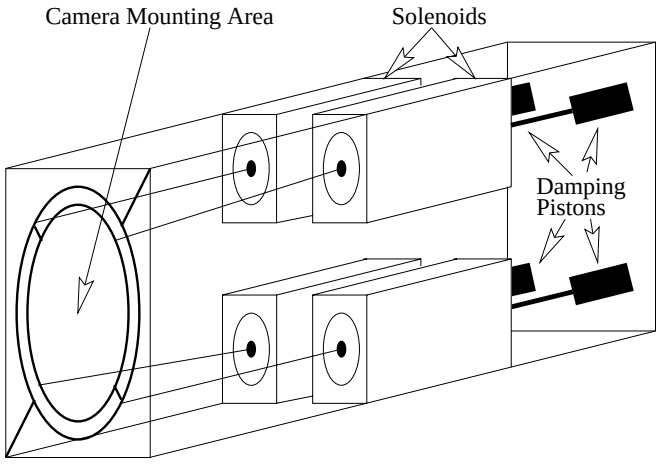


Figure 3.5: Revised schematic design of the mechanical Monocular Active Vision Eye (MAVE) with improved modularity of the key components. This design consists of a gimbal with the camera mounting area, the damping pistons and the solenoids that generate camera movement.

Another unanticipated design complication appeared when testing the operation of the shock absorbers. When a hydraulic shock absorber is compressed or extended, the level of shock fluid in the cylinder changes, due to the displacement by the pistons. The changing level of shock fluid is equalized by air that is separated from the damping fluid by a membrane. The stretched membrane of the shock absorber can assert sufficient force on the shock fluid to re-extend the piston after compression. This is not desirable, as shock absorbers are not intended to play an active role in positioning the gimbal. However, as *opposing actuation* is used to control the movement of the gimbal, one actuator is extended as the opposing actuator is compressed. The length of extension and compression between two opposing actuators is identical for any action they perform. As the shock absorbers are now connected directly to the solenoids, the level of extension and compression between opposing shock absorbers is also identical. Thus, if it were possible to connect the opposing shock absorbers in such a way that the shock fluid could pass freely between them, there would be a closed, self equalizing system with no need for an equalizing membrane.

As the four shock absorbers are now mounted in parallel at one end of the solenoids, the implementation of a pressure equalizing system is possible and benefits from the new modular MAVE design. A shock absorber mounting backplane was constructed in such a way that the shock absorber cylinders screw directly into it. A channeling system was also drilled into the backplane, allowing shock fluid to pass between all shock absorbers. Figure 3.6 shows the rear assembly of the mechanical MAVE. The backplane is manufactured out of perspex, allowing the level of shock fluid to be monitored. The black circles at the end of each shock absorber are holes which allow shock fluid to pass between the shock absorbers and the perspex backplane. The channeling system in the backplane consists of one horizontal and two vertical channels which

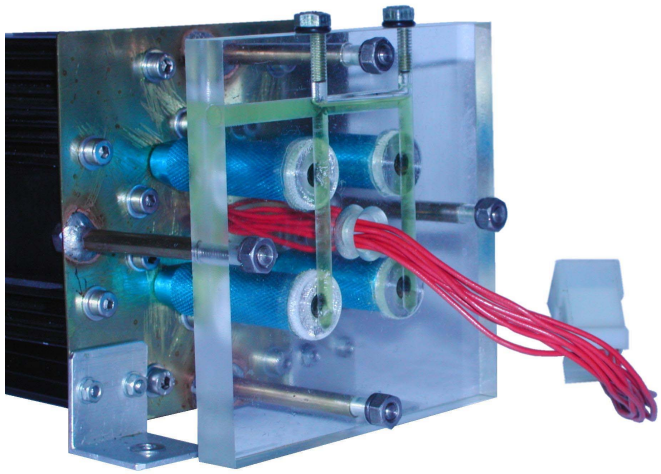


Figure 3.6: Shock absorber assembly with shock fluid channels in perspex. The cables passing through the perspex backplane connect to the solenoids.

connect all shock absorbers with each other.

Apart from providing fixtures and a channeling system for the shock absorbers, the backplane also has to withstand the force applied by activated solenoids. During peak activations this force can be in excess of $14Kg$. To reduce the chance of backplane misalignments as a result of such forces, the stress path was reduced to a small structurally enforced region of the mechanical MAVE. For this purpose, a brass bulkhead was fixed to the heatsinks and solenoids on one side, and to brass spacers on the other side. The spacers extend from the bulkhead and connect directly to the backplane. The effective stress area between the solenoids and shock absorbers is now confined to the area between the bulkhead and the backplane. The stress applied by the solenoids

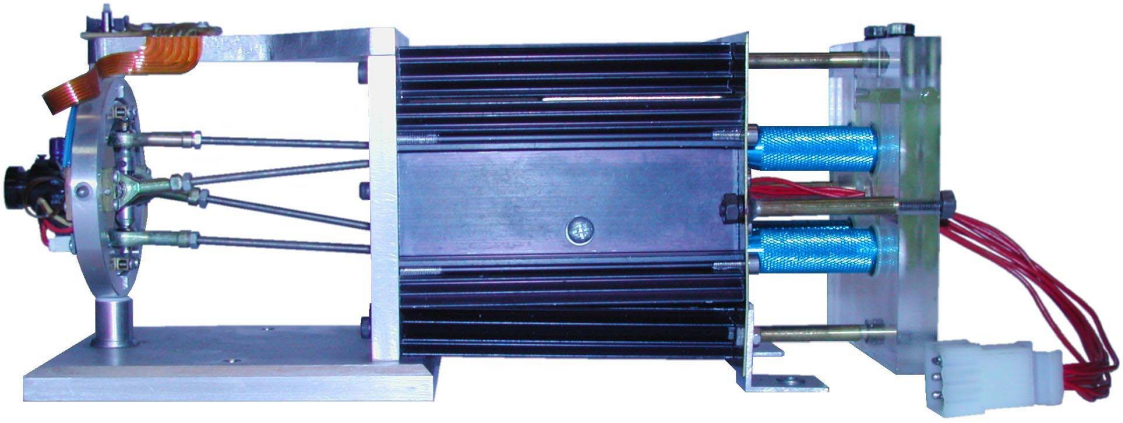


Figure 3.7: Side view of the MAVE mechanics. From left to right, the logical components are: Gimbal with mounted camera, linear actuators under heatsinks and mechanical damping pistons with pressure equalizing channels in perspex.

along the rods towards the gimbal is a linear force and is relayed through the gimbal back to the damping system.

Figure 3.7, shows the completed mechanical MAVE with the new shock absorber fixtures and pressure equalizing system. The gimbal, rod-end bearings, rods and shock absorbers are clearly visible. To protect the solenoids from thermal damage, they have been placed behind black heatsinks.

This completed active vision system has many of the properties that were identified early on in the section and allows the system to be defined as follows:

> "A statically balanced, parallel architecture camera system with two degrees of freedom and feedback controlled, opposing linear actuators."

3.2 Mechanical Target Design

The experimental platform is primarily designed to be used for the investigation of biologically plausible smooth pursuit and saccadic control algorithms. These movements are triggered in humans as a result of analysing complex images, localised excitation within the field of view, auditory stimuli and other influences. To focus the investigations specifically on the sensory-motor control that underlies the movements of interest, the visual data presented to the mechanical MAVE has to be unambiguous and easy to analyse. This is implemented by generating distinct targets in the field of view, which are able to move at continuous speeds, appear or disappear and remain at fixed locations. The control of such targets should be automated, to allow different experimental MAVE controllers to be tested under identical experimental conditions. Automation also reduces the need for human interaction with potentially monotonous experimental tasks.

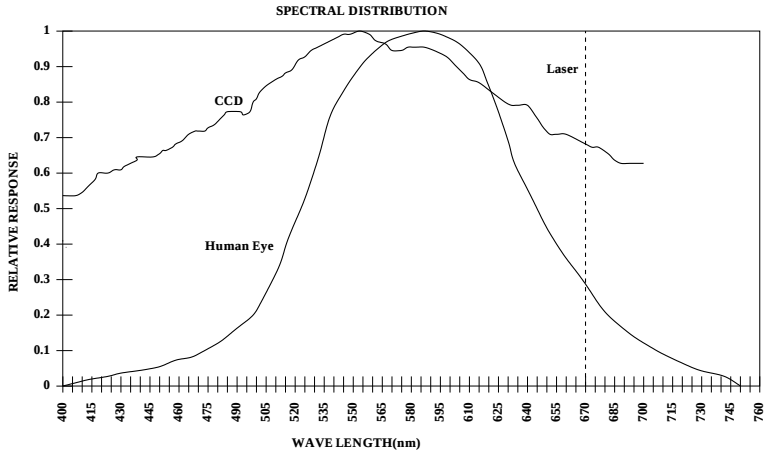


Figure 3.8: Relative response of the human eye and the MAVE CCD camera to the visible light spectrum. The light frequency of the laser targeting system is indicated by the vertical, dashed line.

The retina of the eye and CCD cameras are sensitive to light, figure 3.8, hence a visual stimulation system was envisaged that uses directed light to produce visible targets. This could be simply implemented by projecting light onto a screen, but laser light has properties that are particularly favourable for the test application. Laser beams have a negligible divergence which allows the size of targets to remain virtually unchanged, when moving across a screen. Laser light can be intense enough to produce visible locations on many surfaces and it is available in different wavelengths.

The trajectory of a laser beam can be controlled by mirrors that move in a predefined way. This makes it possible to keep track of exact target locations and make the experimental environment definable and predictable.

Precise clockwise and anticlockwise mirror positions can be controlled by stepper motors. Stepper motors turn with fixed angular resolutions for each step they make. If required, the angular resolution of steps can be changed by adding gearboxes with set ratios. This determines the speed at which the targets can travel across a screen and what distance lies between neighbouring target positions. Determining a gearbox ratio relies on estimating reasonable speed and positioning resolutions. The single limiting factor for this calculation is the stepping speed of stepper motors. Most stepper motors start losing steps at a step speed of $> 450Hz$. A safety margin of $50Hz$ is therefore recommended, restricting the maximum step speed to $400Hz$[6]. If the gearbox ratio is chosen high, the target travels across the screen slowly, but the resolution of possible target locations is high. If the ratio is low, the target travels across the screen fast, but the resolution of possible target positions is low. It is therefore necessary to compromise between the positioning accuracy and the speed at which the target should travel.

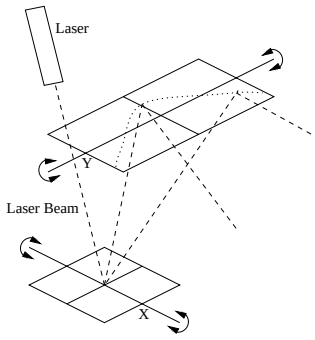


Figure 3.9: Schematic design of the laser target positioning system. Two of many laser trajectories are indicated by the dashed lines. Each of the two surface reflective mirrors rotates in separate directions, enabling the laser target to move in horizontal and vertical directions.

The target system for the experimental set up uses a fixed laser source and two rotating mirrors, figure 3.9. The mirrors are mounted directly on the drive shaft of gearboxes. One mirror controls the horizontal movement of the laser beam, one mirror controls the vertical movement of the laser beam. Strictly speaking, the mirrors do not control exact horizontal and vertical beam trajectories. The movements are slightly curved, due to physical alignment between the mirrors and the laser source and a change from a spherical to a cartesian coordinate system during screen projection.

As the experimental environment is controlled by a computer, it is possible to change the time scale in which operations are performed. This allows experimental MAVE controllers with different computational resource requirements to be run and compared. The target and positioning accuracy can take account of the MAVE controller requirements, allowing the target speed and position accuracy to be adjusted. This means that identical laser target patterns can run at different speeds, depending on the computational requirements of the MAVE control program.

[6]The step speed is determined in software and requires computer dependent constants to be selected.

The calculation of gearbox ratios also takes account of this. This means that a stronger emphasis can be put on deriving a sensible positioning resolution for the targets, instead of trying to increase target speeds. A sensible positioning resolution reflects the mechanical error of the motors and gearboxes. This is a threshold that marks the boundary between the smallest possible positioning accuracy and the positioning error. The manufacturer of the gearboxes and stepper motors used here does not provide performance information that specifies the error margins on backlash and repeatability. Backlash error margins found on high precision gearboxes are often $< 0.15^{\circ}$, but it is not financially viable to use high precision gearboxes to position the laser mirrors here. However a step resolution of 0.15° would allow targets to be positioned at a distance of $0.11cm$ or move at a speed of $44.5cm/s$. This assumes a spherical screen with a radius of $42.5cm$ and the laser mirrors rotating in the origin. In the experimental environment, figure 3.1, the projection is onto a flat screen from a distance of $42.5cm$. This means that the target speed and positioning resolution vary at different locations on the flat screen. The positioning accuracy is in the range $0.11cm$ to $0.62cm$ and the target speed lies between $44.5cm/s$ and $247.8cm/s$.

Assuming 0.15° as a mirror positioning accuracy, it is easy to calculate the resolution of the required gearboxes and stepper motors. In this experimental set up, the stepper motors have a resolution of 7.5° and use reduction gearboxes with a ratio of 50:1.

3.3 Electrical Control Unit Design

The control, calibration and measurement of the mechanical components is performed as a result of control signals being sent to and from a computer. Digital signals provided by a standard computer interface, such as the serial or parallel port, are, however, not powerful enough to control some of the actuators used here. There are also not enough channels to control all the actuators individually. The control unit bridges the gap between the computer and the mechanical components that make up the experimental environment. To perform this task, the control unit establishes a digital communication with the computer at one end and analogue, digital or pulsed communications with the laser scanner and mechanical MAVE at the other end. Industrial hardware standards exist that define ways in which periphery should be connected and controlled through the serial or parallel port of a computer. Many of these standards are implemented in hardware and are integrated into fully functional interface boards that can be used to control external devices.

The control unit is built around one of these interface boards (P8000'2). It provides a digital communication port to the computer at one end and digital and analogue input / output channels at the other end. The digital and analogue channels can be controlled individually and have a higher switching current than the computer interface. For some of the hardware requirements, these currents are still not sufficient and need further amplification.

The communication speed between the computer and the interface board is sufficiently high to support the implementation of control signals that control the solenoids or the stepper motors directly. This does, however, carry a communication overhead that could otherwise be devoted to solving high level control algorithms for the calculation of laser trajectories or activation times for the solenoids. Task specific hardware is therefore integrated into the control hierarchy to drive the stepper motors and high current solenoids. The control unit uses three stepper motor controllers (SMCs) that connect with the P8000'2 and control the stepper motors in the laser scanner. By

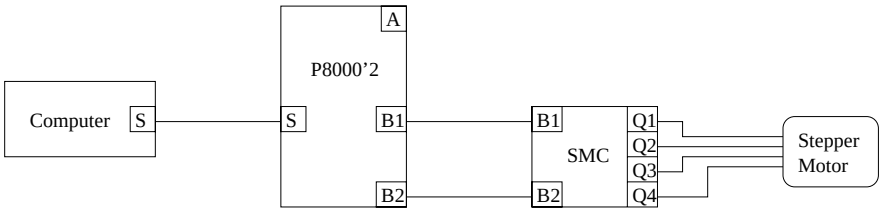
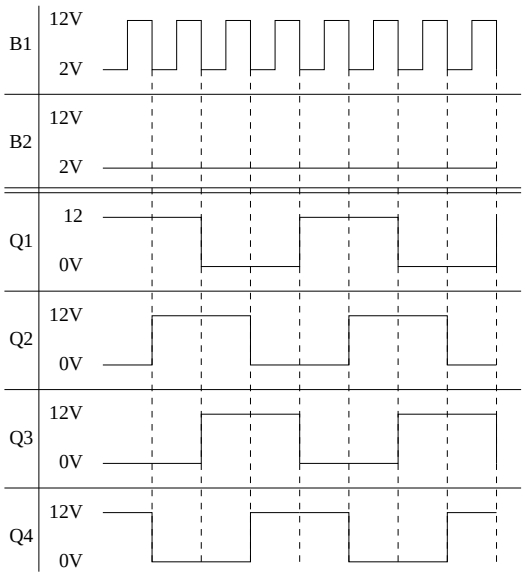


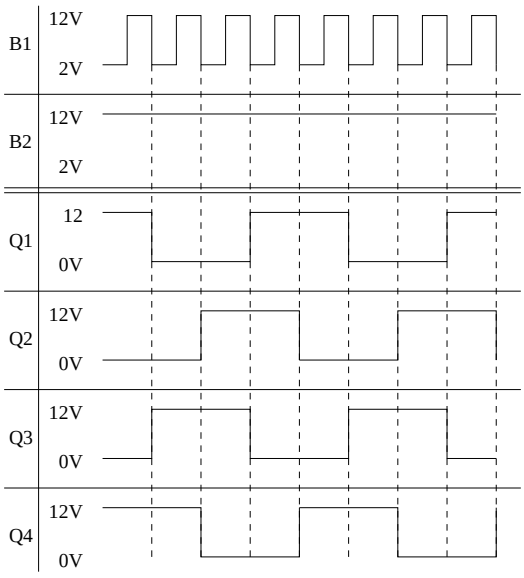
Figure 3.10: Diagrammatic stepper motor control signal hierarchy. *S* is a bi-directional, serial communication channel between the computer and the interface board (P8000'2). *B1* and *B2* are uni-directional, binary control channels from the P8000'2 to the stepper motor control board (SMC). *Q1*, *Q2*, *Q3* and *Q4* are uni-directional, binary control channels from the SMC to the stepper motor.

keeping the SMCs in the control unit, the laser scanner does not require a separate power supply and can be plugged directly into the control unit.

Figure 3.10 illustrates a simplified hardware setup between the computer, the P8000'2, one SMC and one stepper motor. The computer connects to the P8000'2 through the serial channel *S*[7], the interface board connects to the SMC through two binary channels *B1* and *B2* and the SMC connects to the stepper motor through four binary channels *Q1*, *Q2*, *Q3* and *Q4*. *B1* sets each step to be taken and *B2* sets the rotation direction of the stepper motor. These two bit channels are



Clockwise stepper motor rotation.



Anticlockwise stepper motor rotation.

Table 3.1: Signal conversion table that illustrates the functionality of the SMC boards. The binary input signals *B1* and *B2* are set low at *2V* and high at *12V*. The stepper motor control signals are set low *0V* and high at *12V*.

converted by the SMC into four stepper motor drive channels. Table 3.1 show a signal conversion table for the incoming channels *B1* and *B2*, and the outgoing channels *Q1*, *Q2*, *Q3* and *Q4*.

[7]The communication over connection *S* follows a standard I^2C protocol and is not discussed here.

The additional hardware frees computational resources and simplifies the stepper motor control. The use of this dedicated hardware also reduces the risk of skipping steps, due to possible timing errors in the I^2C protocol.

The control unit now contains one port that allows it to communicate with the computer, three ports that drive stepper motors and further digital and analogue input / output channels that can be connected to other components. For simplicity, it should be assumed that there are sufficient channels to allow the integration with all other robotic components.

The following table shows the core components have been used to build the control unit. A corresponding circuit diagram is included in appendix B.

PSU	Main stabilised and regulated power supply unit. This unit supplies **SMC 1** - **SMC 3** and the **Laser Scanner**.
P8000'2	Computer interface board with two integrated power supply units. This board controls the communication between the **Computer, Laser Scanner** and the **MAVE**.
SMC 1, SMC 2, SMC 3	Stepper motor control boards. These units control the step frequency and step direction of stepper motors within the **Laser Scanner**.
4.7 k	4700Ω resistors. Used to covert signals from the **P8000'2** into an LSL-level signal (slow interference-proof logic family).
BUZZER	Piezzo buzzer. This buzzer can give acoustic warning signals.
VIDEO OUT	Phono output port. This port is connected to the camera in the **MAVE** and passes the signal to the frame grabber in the **Computer**.
MAVE PORT	25 pin male D connector. The connection port to the **MAVE**. Pins are mapped one to one.
LASER PORT	25 pin female D connector. The connection port to the **Laser Scanner**. Pins are mapped one to one.

Table 3.2: Electrical modules of the control unit.

3.4 Electrical Target Design

The electromechanical components of the laser scanner connect, through a cable, directly to the electrical components in the control unit. There is no need for intermediate devices in the laser scanner that convert or amplify signals. The stepper motors are mapped onto the three SMCs in the control unit and the laser is an encapsulated unit that can be connected directly to a power supply. The laser is switched on and off by a digital port on the P8000'2 interface board. Photo transistors are connected to analogue input channels on the P8000'2 and are used to calibrate the mirror positions. The calibration is performed by a software controlled sequence, during which the laser is switched on and the mirrors are rotated. The laser is reflected directly onto the photo transistors, under specific mirror constellations. These events are registered and are used to drive the mirrors to default positions.

A physical shutter is mounted on the shaft of a stepper motor, enabling the laser targets to appear and disappear rapidly, without having to switch the laser on and off. The position of the laser shutter is calibrated by using an optical switch, which operates in a similar way to a light

barrier. As soon as the shutter is moved into the light beam, an event is registered and the shutter is driven to a shut position. As the position of the mirrors and the shutter are not retained within the hardware, it is necessary to calibrate their position each time a new control program is started.

The following table shows the core components that were used to build the laser scanner. A corresponding circuit diagram is included in appendix B.

PT1, PT2	Photo transistors. These are used to automate the laser mirrors position calibration.
OS	Optical switch. This switch is used to calibrate the shutter position.
SM1	Stepper motor. This stepper motor controls the laser position along the horizontal axis.
SM2	Stepper motor. This stepper motor controls the laser position along the vertical axis.
SM3	Stepper motor. This stepper motor controls the shutter position.
1 k	1000Ω resistor. Reduces the voltage on the **OS** LED.
LASER	Laser diode with starter. This laser is classed 3*A*. The laser can only be activated through the **P8000'2** control board, which is located in the **Control Unit**.
FAN	Fan. The fan cools the stepper motors.
LASER PORT	25 pin male D connector. The connection port to the **Control Unit**. Pins are mapped one to one.

Table 3.3: Electrical modules of the laser scanner.

3.5 Electrical Design of the Monocular Active Vision Eye

This section looks primarily at how the hardware control and solenoids are integrated into a working system. This covers the design of electrical hardware that is capable of regulating the energy requirements of solenoids and looks at the problems of electro magnetic fields (EMF) and stiction[8].

As solenoids play a key role in the mechanical MAVE, it is important to understand how they operate and how they can be controlled. This understanding is not only vital to the physical design, it also aids the understanding of how and why the MAVE control software can be implemented.

Solenoids use electromagnetic forces to generate motion and apply their full force when switched on and no force when switched off[9]. Varying forces are exhibited when regulating the voltage or using pulse width modulation (PWM). Voltage regulating electrical circuits are primarily used for low current consumers and are often implemented in laboratory power supplies. During regulation, excess energy is usually dissipated in the form of heat, which means that high current consumers require more energy dissipation than low current consumers. PWM has the advantage that the voltage does not change, power is purely switched on and off at variable time intervals, with no need to burn off excess energy. Solenoids consume significant amounts of energy during operation and hence put high demands on the power supply. It is therefore reasonable

[8] Static friction.

[9] Some solenoids are fitted with permanent magnets or springs to automatically retract the piston, after the power has been switched off.

to implement PWM for the control here. PWM also has other electrical design benefits, which are covered later in this section.

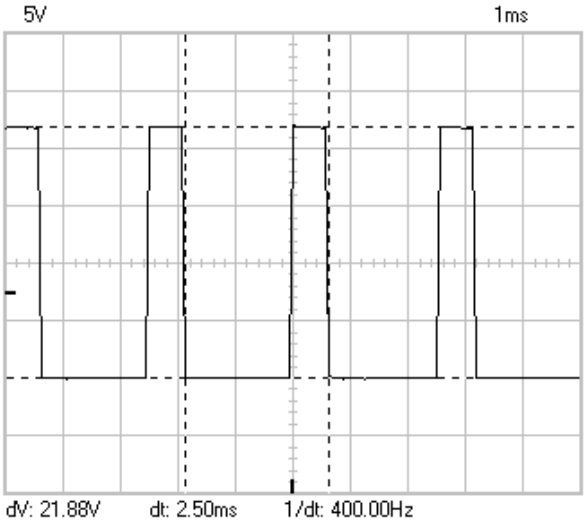


Figure 3.11: Digital storage oscilloscope measurement, taken during the activation of a solenoid with a low pulse width (PW) setting. Horizontal units are in *1ms*, vertical units are in *5V*. The dashed lines enclose the signal boundaries of one pulse and determine the values Δt (*dt*) and and ΔV (*dV*). *1/dt* indicates the pulse frequency.

Before concentrating on the hardware implementation of PWM, it is necessary to understand the underlying concept, shown in a typical oscilloscope measurement in figure 3.11. This illustrates the signal components and characteristics that play a role in controlling the solenoids. The $5V$ in the top left corner refers to the gain in the vertical direction[10]. The $1ms$ in the top right corner refers to the gain in the horizontal direction[11]. The pair of dashed horizontal lines refer to dV (ΔV) and encloses the voltage range of the pulses. In this example, the range encloses $21.88V$. The pair of dashed vertical lines refer to dt (Δt) and enclose the time of one pulse[12]. In this example one pulse encloses 2.5 boxes, which gives a time of $1ms * 2.5 = 2.5ms$. The pulse frequency is given by $1/dt$ with $400Hz$. The values dV, dt and $1/dt$ are constant and can only be changed by physically modifying the robotic hardware.

To prevent confusion, it is important to distinguish between the terms pulse and pulse width (PW): A pulse is always $2.5ms$ (dt) long, which means that a sequence of pulses runs at a frequency of $400Hz$ ($1/dt$). The PW on the other hand is variable and relates to a fraction of dt, during which the voltage $21.88V$ (dV) is switched on. As dt is $2.5ms$ long the PW cannot exceed $2.5ms$. This means that a PW of $0ms$ would generate a constant signal of $0V$, and a PW of $2.5ms$ would generate a constant signal of $21.88V$. Intermediate PW values would generate signals, similar to the one seen in figure 3.11. In the following context, the PW will be used in unit size percent, where a PW of $0ms$ relates to 0% and a PW of $2.5ms$ relates to 100%. As the PW increases from 0% to 100%,

[10] The height of each box accounts for $5V$.
[11] The width of each box accounts for a time period of $1ms$.
[12] This is not the pulse width.

the relative time during which the voltage *dV* is switched on also increases. This means that the average magnetic field in the solenoids can increase and apply a stronger force on the solenoid plungers.

Similar to the control of the stepper motors in section 3.4, implementing PWM as a result of timed control signals by the computer would carry a significant communication overhead. It is more elegant to send control command from the computer to some hardware that is capable of generating a high current PWM signal. This reduces possible communication errors and frees computational resources.

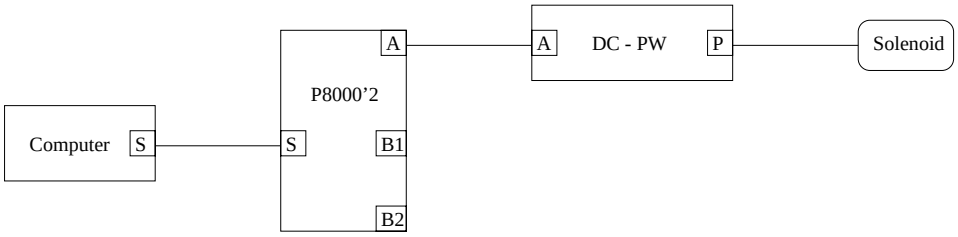


Figure 3.12: Diagrammatic solenoid control signal hierarchy. *S* is a bi-directional, serial communication channel between the computer and the interface board (P8000'2). *A* is a uni-directional analogue control channel from the P8000'2 to the direct current to pulse width modulator (DC - PW). *P* is a high current, uni-directional, pulse width modulated (PWM) control channel from the DC - PW to the solenoid.

Figure 3.12 illustrates a simplified hardware configuration between the computer, the P8000'2, a direct current to pulse width modulator (DC - PW) and a solenoid. The computer connects to the P8000'2 through a serial channel *S*. The interface board connects to the DC - PW board through an analogue channel *A* and the DC - PW board connects to the solenoid through a high current digital channel *P*. *A* carries a control voltage between 0*V* and 10*V* in 65 voltage steps. There is one step between 0*V* and 2*V*, and 64 steps between 2*V* and 10*V*. 0*V* causes no signal to be sent to the solenoid. Values between 2*V* and 10*V* cause the DC - PW to generate PWs on *P*, which have

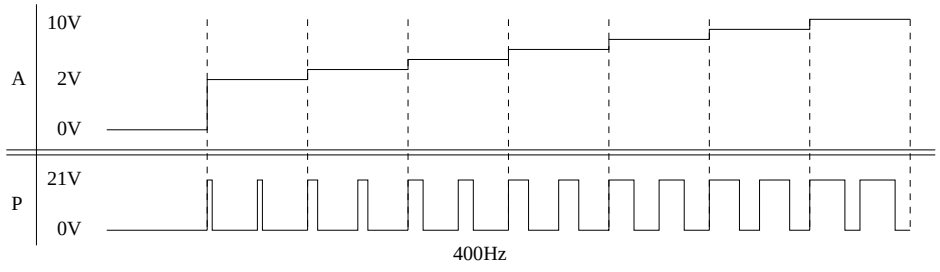


Table 3.4: Signal conversion table that illustrates the functionality of the DC - PW boards. The analogue input signal *A* is set low at *0V* and high between *2V* and *10V*. The high range can be set in steps of *0.125V*. The solenoid activation signal *P* is set low at *0V* and high at *21V* with a modulated PW.

a PW in the range 10% to 70%. Table 3.4 illustrates the correspondence between the input signal *A* to the DC - PW and the output signal *P* to the solenoid.

Once the high current solenoid control hardware is implemented it may be assumed that the next step is to find a solenoid and connect it. This is correct, but solenoids are not normally controlled by PWM. They are usually only switched on or switched off, using an AC or DC supply. This means that it is necessary to carefully identify a solenoid that can apply a sufficient force for this specific application but also have a long life expectancy.

Solenoids have non-linear activation curves, with respect to the stroke of the solenoid plunger. They can also run at different *duty ratings*, which specify for how long they may be switched on, during a given time interval. The duty rating is derived by a number of parameters, such as the voltage and current at which the solenoid will be run and the degree to which the plunger is retracted into the magnetic coil[13]. Each solenoid in the mechanical MAVE is run at a duty rating of 8%. This means that for a given time interval, the solenoids should only be switched on for 8% of the time. This is a very short time indeed and one could argue whether or not this time is sufficient for the application. In the mechanical MAVE, the solenoids are activated for very short times, during saccades and smooth pursuit. Even with the short time periods that saccades and smooth pursuit take, 8% may still appear rather unrealistic. However, if this time is moved into perspective and the way in which PWM works is considered, 8% becomes plausible.

The DC - PW hardware has been set to permit PWs in the range from 10% to 70%[14]. So if the solenoids were switched on permanently at any PW between 10% and 70%, they would still exceed their permissible duty rating of 8%. However, there are many other factors that have to be considered, which reduce the time during which each solenoid is switched on. First of all the system uses pairs of solenoids, which means that at least two solenoid are switched off, while the opposing solenoids are switched on. Even with continuous diagonal camera movements, the activation of each solenoid within the system is halved, reducing the maximum activation time from 70% to 35%. Between sequences of solenoid activation, the computer also has to change the position of the laser target. At maximum target speed and a PW of 70%, the MAVE is at least twice as fast as the laser target. As the target and the MAVE cannot be controlled at the same time[15], MAVE activations are switched off at least twice as long as it takes for the MAVE to move to the new target position. This means that the solenoid activation time can again be reduced, this time by $\frac{1}{3}$, reducing the average on time from 35% to 11.6%. From the mechanical aspects alone, the expected maximum solenoid activation time is just 3.6% above the permissible rating. The computational requirements have not yet been added, but it is assumed that image processing, and communication overheads will add extra delays that reduce the activation time even further, approaching the permissible duty rating of 8%. This means that solenoids run at a duty rating of 8% are appropriate for the application in this experimental environment. The solenoids are only switched on for very short time periods, preventing them from overheating, but also allowing them to be driven with a sufficient current that enables them to apply a high force to the camera platform, figure 3.13.

[13]The calculation of duty ratings is not be performed here. It is sufficient to know what a duty rating is and how the electrical control of this environment affects it. Further information on duty ratings can be found in [105].

[14]The PW and duty rating are values that can be directly compared with each other, as they both define a percentage during which electricity flows to the solenoids.

[15]Parallel scheduling of the mechanical MAVE and the laser target would have been too complex to implement here, with respect to the real-time requirements of the hardware control.

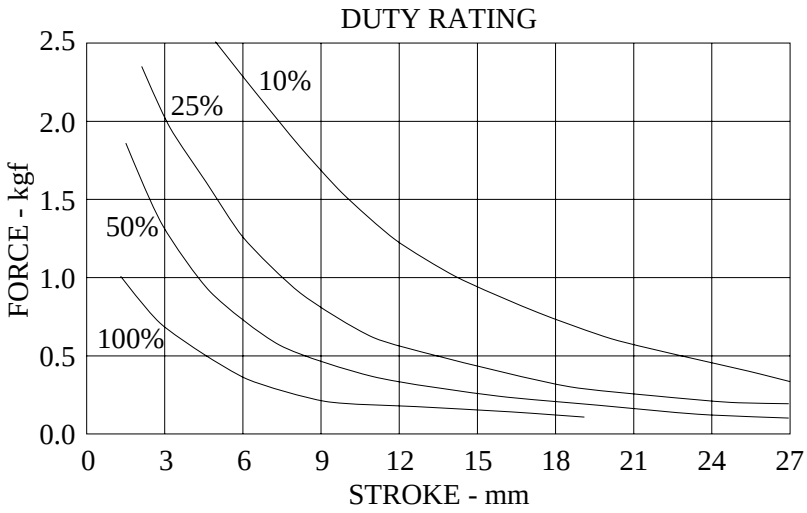


Figure 3.13: Non-linear activation curves for the solenoid force at varying piston strokes. The four curves represent manufacturer defined performance values for different duty ratings. The MAVE is designed to operate at solenoid duty ratings of *8%*.

During the design of the mechanical control hardware and the integration of the solenoids, it is important to consider the effects of electro magnetic fields (EMF), especially as high currents are switched on and off at very short time intervals. EMF occurs in every electrical application, but often these magnetic fields only have a negligible impact on surrounding components and the environment. The control of coils by PWM in the mechanical MAVE is, however, a potential source of interference that can cause problems. It is therefore important to implement safety precautions to prevent interference with the control hardware.

Interference with the control and other electrical hardware can manifest itself by radiating electromagnetic fields that build up and break down when coils are activated and deactivated. These changing magnetic fields can induce an electric current into a conductor passing through them. Negative effects on the control hardware, as a result of electromagnetic fields, can be reduced by added shielding and increasing the distance between the solenoids and the control hardware. High-current diodes protect the PW modulators from the large reverse voltages induced into the coils by the collapse of their electromagnetic fields when they are turned off. Electrical interference with the CCD camera is reduced by using a second, independent power supply that only supplies electricity to the camera. Ferrite rings and capacitors are also used as low-pass filters that block high frequency interference with the camera image.

The last design issue considered during the design of the mechanical MAVE focuses on the hydraulic damping system. Hydraulic systems in general, used as actuators or shock absorbers, suffer from a phenomenon known as stiction. This is an initial “stickiness” that occurs when pistons are activated and static friction is greater than kinetic friction. In the case of the mechanical MAVE, stiction can lead to unwanted jumping between camera positions. The effect of stiction can be reduced by vibrating the hydraulic system. The PWM, controlling the solenoid activation, can be adjusted to run at varying frequencies, allowing the solenoids to resonate in a definable way. The solenoid plungers are rigidly connected to the damping pistons of the hydraulic shock

absorption system allowing the solenoid resonance to be applied directly to the pistons of the shock absorbers. The PWM can hence be adjusted to apply resonant stiction reduction. However, as the pistons of the shock absorbers are not accessible when integrated into the MAVE, it was not possible to measure the piston resonance with a stethoscope. Vibration tests on purpose built test benches were also ruled out due to the risk of damaging other mechanical components. It was hence decided to adjust the PWM frequency to lie within the mid range of the possible PWM scale, $400Hz$. Measurable camera jumping or inaccurate camera positioning does not appear to occur at this frequency. IBM had similar stiction problems with their RS-1 (Model 7565) robot [52, 135]. They also used a frequency of $400Hz$ to overcome the problem of inaccurate positioning.

APS 200	High current AC/DC converter. This board provides a stabilised, unregulated DC input to the **DC - PW** modulators.
T1	High current AC/AC transformer. This transformer supplies $15V$ at $20A$. AC input to the **APS200**.
EF-2	AC/DC converter. This board produces a stabilised and regulated $12V$ output to the **CAMERA** module.
T2	AC/AC transformer. This transformer supplies $15V$ at $0.3A$. AC input to the **EF-2**.
PSU min 2A	This is an optional component and is designed to extend this MAVE to be position controlled. This power supply unit should have a regulated and stabilised $12V$ output and would supply the units **SMC 1** and **SMC 2**.
DC - PW 1, DC - PW 2, DC - PW 3, DC - PW 4	DC/PWM. These units convert a low current DC input into a pulsed, high current output. The maximum output is rated at $6A$. The pulsed output controls the actuation of the solenoids **S1-S4**.
Camera	Miniature camera with an interchangeable lens. The camera produces a standard video output at a frame rate of $50Hz$.
SMC 1, SMC 2	These are optional components and are designed to extend this MAVE to be position controlled. These *SMC*s are identical to the ones found in figure B.2.
4.7 k	These are optional components and are designed to extend this MAVE to be position controlled. 4700Ω resistors. Used to covert binary control signals into LSL-level signals. These resistors must be included when using the standard stepper motor controllers.
D1, D2, D3, D4	These diodes are used for extra circuit protection.
P1, P2	These are optional components. The potentiometers can give a direct feedback on the horizontal and vertical rotation of the MAVE. This is an electrical implementation of muscle spindles.
S1, S2, S3, S4	Linear actuators. These solenoids apply forces in a linear direction. At continuous rating they have a power consumption of $10W$.
SM 1, SM 2	These are optional components and are designed to extend this MAVE to be position controlled. Stepping linear actuators, based on the principle of stepper motors.
MAVE PORT	25 pin female D connector. The connection port to the **Control Unit**. Pins are mapped one to one.

Table 3.5: Electrical modules of the mechanical MAVE.

Table 3.5 shows the core components that were used to build the MAVE. A corresponding

circuit diagram is included in appendix B.

3.6 Hardware Performance and Calibration

The performance of most active vision systems is specified by acceleration, velocity, angular resolution and angular range, but most of the literature reviewed for this research draws no comparison with the performance of the human eye. As the system designed here is based specifically on properties of the human eye, it is only reasonable to draw comparisons.

Velocity tests were carried out with the maximum permissible PW of 70% and showed that the mechanical MAVE can reach a speed of $930^{o}/s$. This is very close to the reported peak human velocity of $900^{o}/s$ [28].

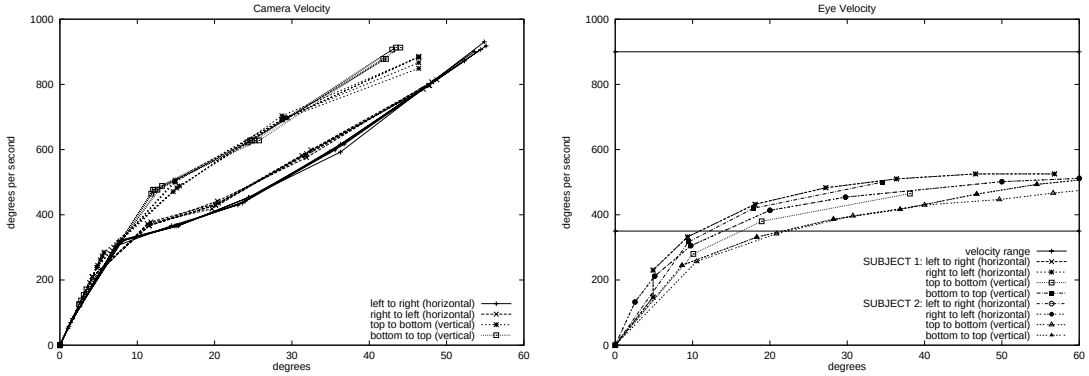


Figure 3.14: The left graph shows the maximum angular velocity for horizontal and vertical camera movements generated by the MAVE. The acceleration is continuous until the physical end position is reached. The right graph shows the maximum angular velocity of horizontal and vertical long human saccades (two subjects). The velocity decreases when approaching the target. The horizontal lines mark the difference in reported maximum human eye velocity, measured in a number of individuals. The angular rotation is displayed along the X axes and the velocity of$^{o}/s$ is given along the Y axes.

Figure 3.14 compares the velocity graphs of the mechanical MAVE with velocity graphs of the human eye. The experimental conditions for each graph were slightly different, although the minimum and maximum saccadic velocity range of the human eye is specified by the two horizontal lines. The camera velocity curve, generated by each solenoid was measured five times and plotted. There was no target onto which the camera was to foveate. Instead maximum actuation was continuously applied to move the camera from one mechanical end position to the opposite one. In the velocity graph of human eye movement, two subjects were instructed to perform long saccades to defined positions. A difference can be observed between the saccadic performance of the two individuals, although horizontal and vertical behaviours are very similar. The performance of the mechanical MAVE, on the other hand, shows large deviations between horizontal and vertical movements. This was to be expected and is due to the physical properties of the gimbal. During vertical movements, actuators move the inner gimbal ring. During horizontal movements, actuators move the inner and outer gimbal rings. This mechanical asymmetry can cause problems for controllers that are not able to represent these asymmetric properties. This is for example the

case with the adaptive and non-adaptive controllers that are introduced in more detail in chapter 5. These controllers would generate asymmetric camera positioning with target overshooting in the vertical direction and undershooting in the horizontal direction. These mechanical properties can be partially compensated for by the following software measure: If one assumes that controllers generally control the activation time, but have no influence on the PW, it would be possible to set individual PW values for each actuator that moves the gimbal. By setting a higher PW for horizontal activation than for vertical activation, it should be possible to achieve a more balanced gimbal positioning behaviour.

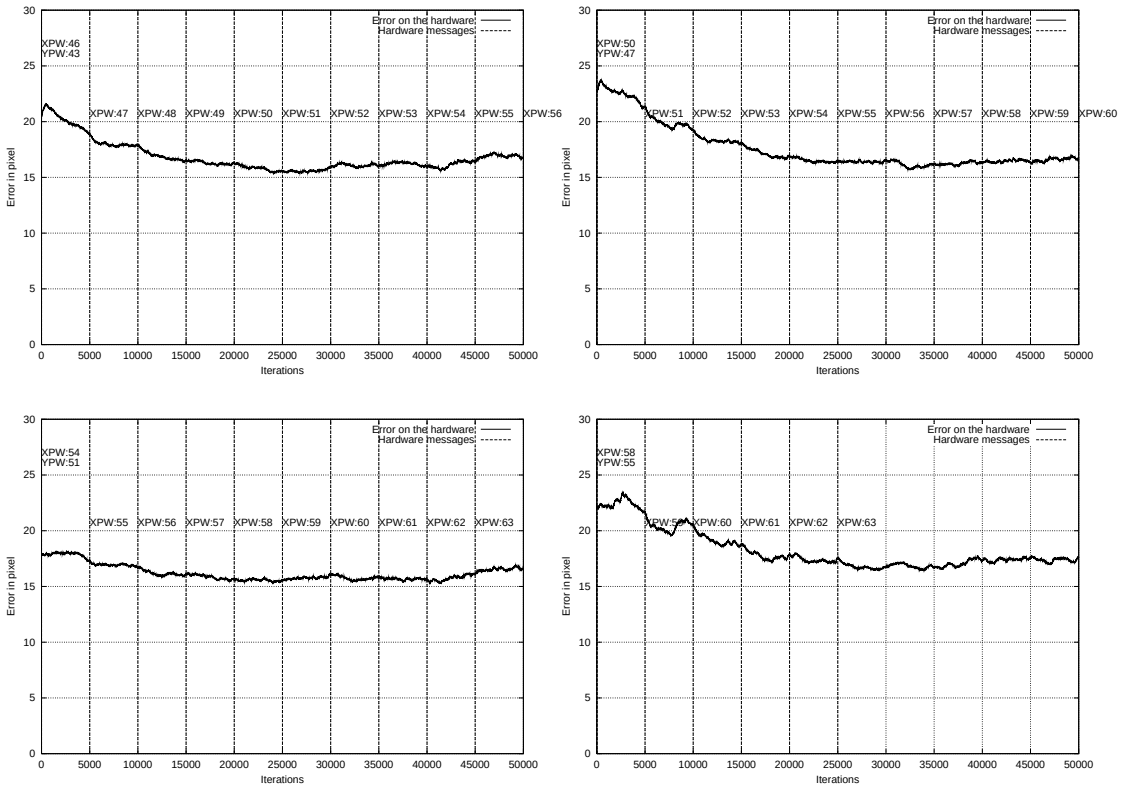


Figure 3.15: PW calibration graphs for the software adjustment of horizontal and vertical PW values that compensate for the asymmetry of the mechanical gimbal dynamics. Each of the four graphs was generated by running the adaptive saccadic controller for 50000 iterations on the robotic hardware. The vertical PW values were fixed during each trial to: 43, 47, 51 and 55. The horizontal actuator PW was incremented by 1, every 5000 iterations. Optimal PW values for the horizontal and vertical activations appear at graph locations with minimum error: XPW:50 / YPW:43, XPW:54 / YPW:47, XPW:58 / YPW:51 and XPW:62 / YPW:55. All four graphs were smoothed by recursive averaging with a weight of 0.002.

The above calibration tests determine four optimal PW pairs that can be used to compensate for the asymmetric properties of the gimbal. The graphs also confirm the possibility of the software compensation measure. Incrementing PW values are applied to the horizontal actuators, with respect to a fixed PW value for vertical actuators. These tests are run on the adaptive saccadic controller, defined in chapter 5. This controller tries to minimize the error between target positions and the camera fovea, by applying a uniform activation factor to all four actuators, in accordance

with an error function. If the positioning characteristics differ between horizontal and vertical actuations, the positioning error increases. Figure 3.15 illustrates the process by which horizontal PW settings are derived for a given vertical PW value. It can be seen that the pixel error is lowest for horizontal PW values that are a value of seven greater than the vertical PW activation values.

This software compensation measure is not a perfect mechanism, but it is a simple measure that enables a more symmetric control of the camera platform. PW pairs generated by these tests are used in all following controller tests of this dissertation, including the simulator.

Nonstop trials have shown that the system can run reliably for durations in excess of 30 hours, with a count of camera activations in excess of one million and target mirror activations in excess of ten million. These trials also show that the solenoids, with a very low duty rating, are activated for sufficiently short periods that prevent thermal damage. It is also shown that tests can be run for extended periods without the need for experiment supervision or intervention. However it was also observed that fluctuations in the ambient temperature have an effect on the performance of the active platform. As the ambient temperature increases, the mechanical MAVE appears to "fatigue" and movements become slower and less controllable. This behaviour was measured by running three temperature trials in the ranges: $15 \pm 1^{\circ}C$, $26 \pm 1^{\circ}C$ and $34 \pm 1^{\circ}C$. During each trial, the software conditions were identical. The laser target locations were generated with the same random number seed and the PW was changed every 12000 iterations, without informing the non-adaptive saccadic MAVE controller, discussed in chapter 5. This controller is particularly well suited for these tests, as it is a mathematically simple controller and does not try to adapt or compensate for changes in the environment. This means that any change in the environment should have a significant impact on the evaluated performance of the MAVE controller.

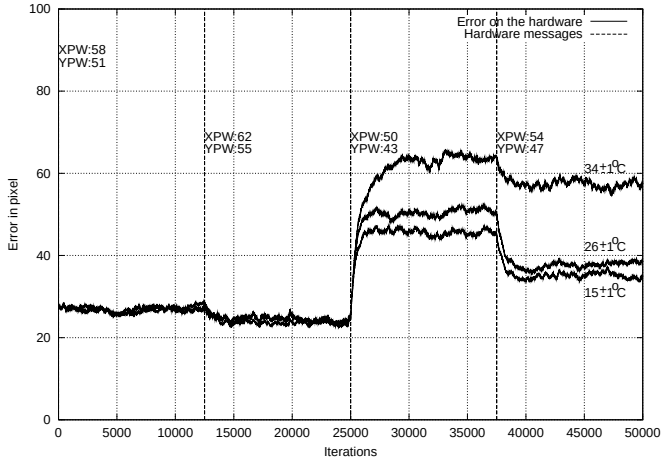


Figure 3.16: Error graphs that demonstrate the effect of ambient temperature on the performance of the mechanical MAVE. The graphs were generated by running three trials with the non-adaptive saccadic controller under identical target positioning sequences. For each trial the ambient temperature was changed and set to one of three ranges: $15 \pm 1^{\circ}C$, $26 \pm 1^{\circ}C$ and $34 \pm 1^{\circ}C$. The three curves were smoothed by recursive averaging with a weight of 0.002.

Figure 3.16 shows the error graphs for the three trials. During the first half of the test (0-24999 iterations), all graphs appear to perform similarly well, but when the PW values change from XPW:62 / YPW:55 to XPW:50 / YPW:43, the positioning capabilities start to vary considerably

between graphs. Ambient temperature fluctuations between $15 \pm 1^{\circ}C$ and $26 \pm 1^{\circ}C$ clearly influence the performance of the mechanical MAVE, but the impact is lower than in the temperature range between $26 \pm 1^{\circ}C$ and $34 \pm 1^{\circ}C$. This is the case despite the fact that there is a smaller temperature range between $26 \pm 1^{\circ}C$ and $34 \pm 1^{\circ}C$ than between $15 \pm 1^{\circ}C$ and $26 \pm 1^{\circ}C$. The figure also shows that the two lower temperature graphs start to significantly recover their performance when the PW values increase to XPW:54 / YPW:47, but the high temperature graph does not exhibit such a strong recovery.

There are a range of components and a range of ways these components could be affected by ambient temperature fluctuations. However, it is assumed that the hydraulic damping system is most vulnerable. It may be that the polymer seal rings, fitted inside the shock absorber cylinders expand and contract during ambient temperature fluctuations. This could be sufficient to change the stiction properties between the pistons and the seal rings when the pistons travel in and out of the shock absorber cylinders. As the temperature rises, the seal rings may be expanding at a different rate than the steal pistons. A second possible reason could be due to changes in the damping fluid viscosity. As the temperature of the damping fluid rises it could be assumed that the viscosity of the damping fluid decreases. This means that the damping fluid disperses quicker, allowing the pistons and seal rings to touch under less pressure and increase the friction. During the last 25000 iterations of the trial at $34 \pm 1^{\circ}C$, it is assumed that the friction in the damping system was sufficiently high to virtually prevent the movement of the gimbal altogether.

As a result of these observations, all following tests in this dissertation were run in the ambient temperature range of $22.5 \pm 2.5^{\circ}C$.

3.7 Summary

This chapter introduces the physical experimental environment from the requirements analysis to the implementation and test phase. Many of the details that were part of the development have been omitted to demonstrate the essential thought processes that were involved in designing the system. Some developmental stages were also discussed that led to modifications of the initial design, which as a result had a beneficial impact on the final system. It is shown that a limited budget and an often unconventional approach can be sufficient to develop an active vision platform that has some of the high level properties found in the human extra-ocular system.

The next chapter covers the software development and the simulator of the experimental environment.

Chapter 4

Software Environment and Simulation

The software fulfills a whole range of essential tasks in the experimental environment, such as communicating with the electrical hardware and scheduling between the laser and the Monocular Active Vision Eye (MAVE) control programs. It also provides a development environment that aids the structured implementation and testing of image processing, laser scanner and MAVE control software.

This chapter describes the hierarchy and interaction of all major software components that have been developed and integrated for this project. This starts by introducing the components of

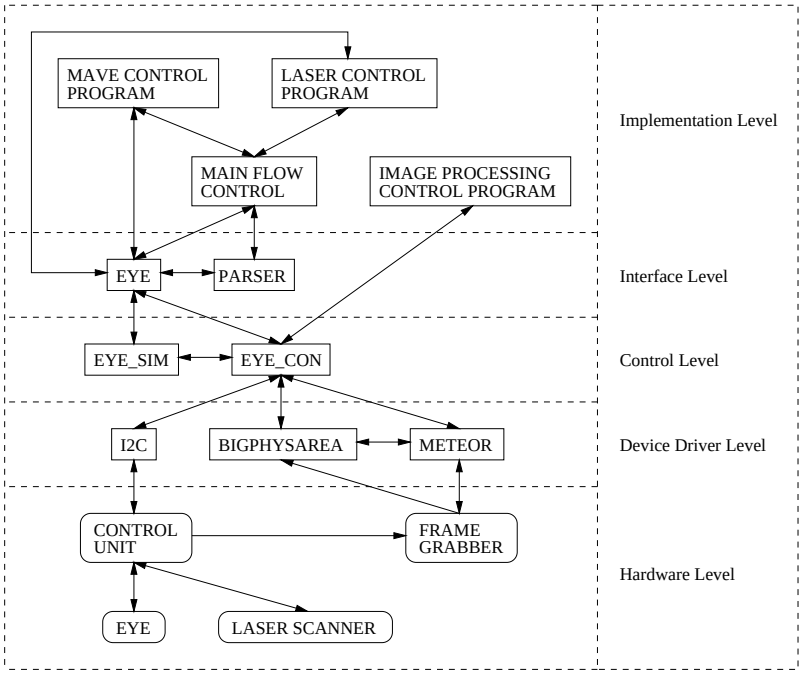


Figure 4.1: Hierarchical data flow diagram of information flowing from the implementation level to the control of the robotic hardware. The left side of the figure shows the logical software and hardware components of the experimental environment. The right side of the figure defines the categories to which the logical components belong.

the hierarchical data flow model in figure 4.1. The simulator, in particular, forms an important part of the development and is covered in detail in section 4.6. This covers mathematical models of the view system and models of hardware properties that had to be simulated.

4.1 Implementation Level

The implementation level is the top interaction level between the user and the experimental environment, allowing image processing, laser target and MAVE control programs to be developed, controlled and tested independently. This level also has facilities that enable the user to control the simulator or hardware directly over the keyboard or through a pipe from another program. For monitoring the interaction and evaluating the performance of experiments, there are also analysis tools that allow generated log files to be manually parsed and analysed. Task scheduling and log generation is also configured from here.

Once top level control programs have been developed and compiled into the software environment, experiments can be run and results processed without intervention by the operator. This makes experiments repeatable, reduces the risk of exposure to laser radiation and allows long sequences of experiments to be run. Some experiments conducted for this dissertation have been running without interruption for periods in excess of 8 hours.

This implementation level utilises four files that control experiments which run on the experimental hardware and three files that control experiments which run in simulation. The use of three / four files at this level aids the modular implementation of experimental components and allows image processing controllers, laser target controllers, MAVE controllers and the experimental test files to be interchanged freely at compile time.

Image Processing Control Program This file is interchangeable and implements image analysis and gaze control that determines where the camera is to point next. The software in this file can be developed independently, without interfering with the MAVE control, laser target control or experiment scheduling and analysis. This file is only used in hardware mode and has no direct interaction with any other file in the implementation level. Hence there is no need to set up scheduling or cycle times. The control level tests the state of returned data to determine whether to use the default system image analysis or the software contained here. The program should interface with the commands described in appendix A.

MAVE Control Program This file is interchangeable and contains the MAVE controller implementation, which moves the camera to the location specified by the *Image Processing Control Program*. The software in this file can be developed independently, without interfering with image processing, the laser target control or experiment scheduling and analysis. The program should interface with the commands described in appendix A.

Main Flow Control This file is interchangeable and contains the instructions that control the operation of experiments. It specifies whether to use the simulator or the mechanical environment and defines the scheduling parameters for the laser scanner and MAVE control programs. This file also contains the log file processing commands that control what information is to be stored for later analysis. The evaluation of the generated data can also be controlled from here. This file also contains information for environment changes, which is used for changing the experimental environment, without informing image processing, the laser scanner or the MAVE control program. This is particularly useful when testing adaptive MAVE control programs that are expected to be able to adapt to changing operational

conditions. The environment changing code and other control commands should interface with the commands described in appendix A.

Laser Control Program This file is interchangeable and contains the control program that guides the laser targets of the laser scanner. The software in this file can be developed independently, without interfering with image processing, MAVE control or experiment scheduling and analysis. The controller should interface with the commands described in appendix A. A coding restriction for the laser controllers requires that control is returned to the scheduler, each time the laser controller has reached the new target position. The return of control allows the scheduler to allocate time to the MAVE control program. The following figures 4.2, 4.3, 4.4 and 4.5 demonstrate test patterns generated by laser target controllers that were implemented as part of the software development phase. These prototype controllers were not used to test the properties of the MAVE controllers investigated in this dissertation. The smooth pursuit and saccadic test controllers are introduced later in chapter 6.

The following images were generated by running the stationary MAVE controller that points the camera at the starting point of the test pattern, where it remains until the laser test pattern has completed. Target locations were plotted each time a new target location was reached. The figures show pairs of images, generated by the same controller. Left images were generated on the mechanical environment, right images were generated in simulation.

Random

This controller produces randomly distributed laser target locations. Definable parameters include the minimum and maximum distance between target locations and the radius of an imaginary circle in which the random locations are distributed. The random pattern can also be set to be repeated, running through the same target sequence several times.

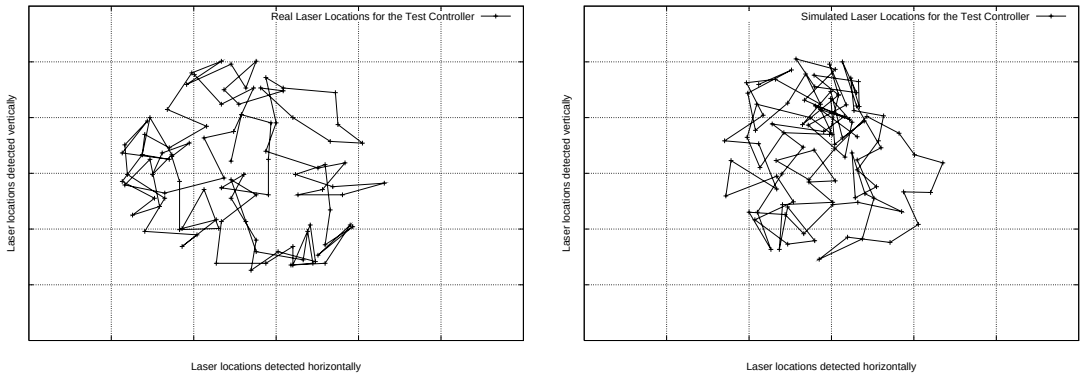


Figure 4.2: Random target positions, generated by a laser target control program and recorded by the MAVE in a stationary position. The left image was produced by the physical robot, the right image was produced in simulation.

Circular

The circular pattern controller generates laser target positions that are located along the line of an imaginary circle. The starting position can be set and the clockwise or anticlockwise direction in which the laser targets are to be positioned. Further parameters are the circle radius, the arc between laser target locations and the number of pattern repetitions.

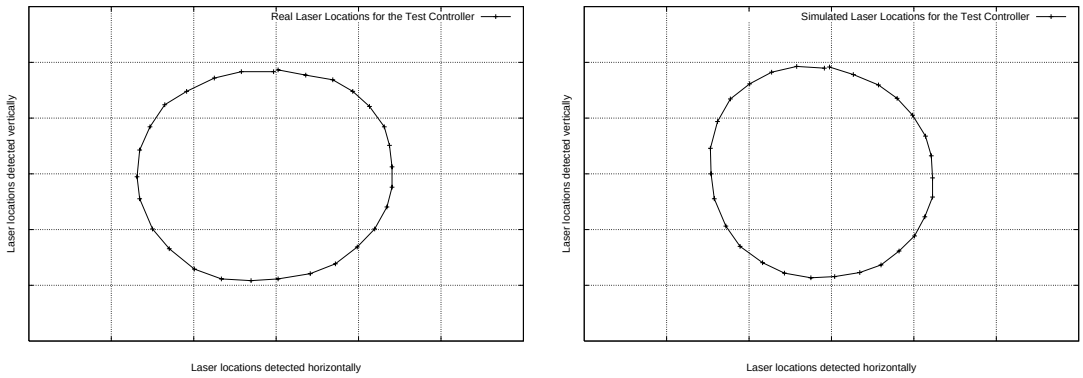


Figure 4.3: Circular arrangement of target positions, generated by a laser target control program and recorded by the MAVE in a stationary position. The left image was produced by the physical robot, the right image was produced in simulation.

Rectangular

The rectangular controller positions laser targets along the lines of an imaginary rectangle, which has a definable size and orientation. The distance between each laser target location and the number of pattern repetitions is also definable.

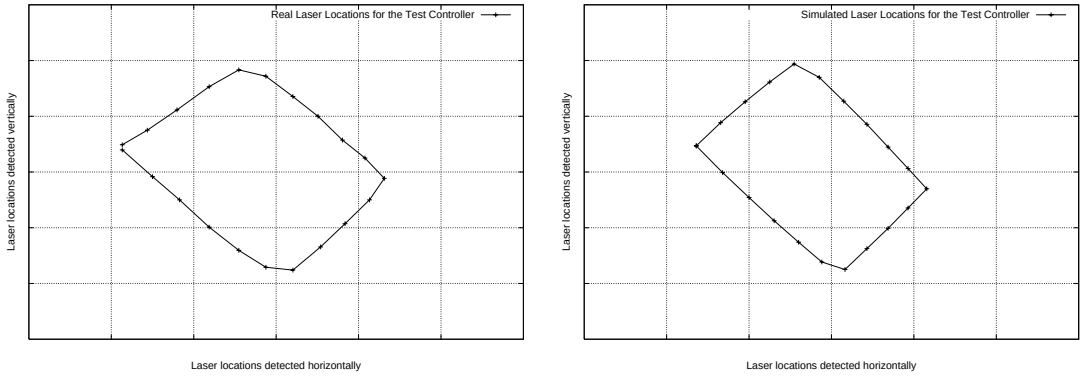


Figure 4.4: Rectangular arrangement of target positions, generated by a laser target control program and recorded by the MAVE in a stationary position. The left image was produced by the physical robot, the right image was produced in simulation.

Parallel

The parallel controller produces laser targets that are arranged along virtual parallel lines. These parallel target arrangements can be generated horizontally or vertically with varying spaces between targets on parallel lines and between the lines themselves. The starting corner and number of pattern repetitions can also be specified.

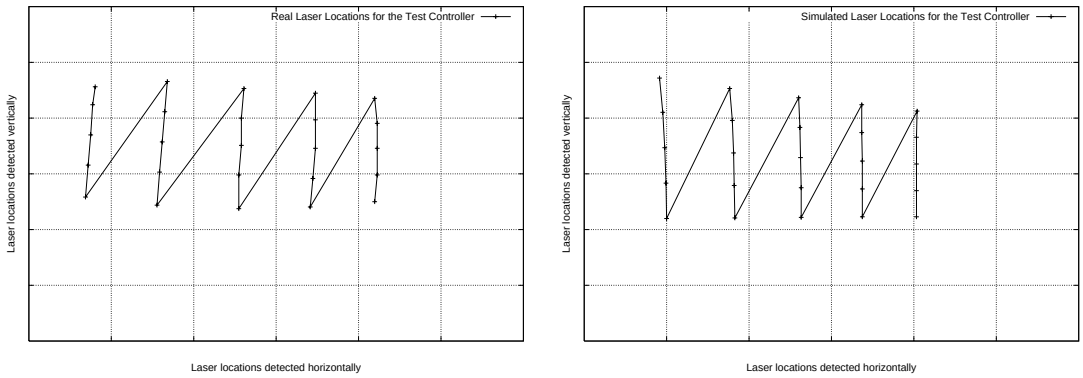


Figure 4.5: Parallel arrangement of target positions, generated by a laser target control program and recorded by the MAVE in a stationary position. The left image was produced by the physical robot, the right image was produced in simulation.

4.2 Interface Level

This level controls the generation and assessment of log files. These are used to store performance information about the MAVE and laser target controllers. This level also forms a common interface to the control libraries of the hardware and simulator. The task scheduler which switches between the laser scanner and the MAVE control programs is also implemented here.

Eye This file forms a meta-library as a front end to the simulator and robotic control libraries. This permits laser and MAVE control programs to run on the simulator or mechanical MAVE, without requiring modification. This modular design paradigm also enables the implementation of test tools that can access both, the simulator and robotic hardware.

Scheduling is also implemented here and can be operated in two ways, either by allocating time slices to the laser and MAVE control, or by allowing a certain number of time independent procedure calls to the laser target and MAVE control. The latter operation does not take account of time constraints and allocates CPU access each time a set number of defined processes have been completed. This allows control programs with varying resource requirements to be scheduled, without having to modify time dependent scheduling parameters. This enables the laser target program to terminate with the completion of the MAVE control program, independent of the MAVE control program resource requirements. This capability is important for the uniformity of resource independent performance analysis. Experiments have shown that the controllers, tested in chapter 6, have a wide range of resource requirements.

A description of all commands accessible in this meta-library is given in appendix A.

Parser The log files, generated during experimental trials, are parsed and evaluated here. Depending on the type of experiments run, and the type of evaluation required, the parsing process can be set to produce results in text or postscript format. This enables rapid printing and inclusion of test results into documents. Results in postscript format are shown throughout this dissertation. These include the laser test patterns shown earlier in this chapter and the error graphs shown in section 4.6.3 and chapters 3 and 6.

The interface specifications for the control of the parsing facility are included in appendix A.

4.3 Control Level

The control level implements the simulator and the hardware control code. These are independent files that share common structures and constants, enabling the interchangeability of the hardware and simulator control. The simulator is modelled on physical properties of the laser scanner and mechanical MAVE.

The implementation of the simulator is discussed in greater detail in section 4.6.

Eye_con The hardware control primarily converts calls from the interface level into device driver calls. Some range checking and processing is performed to prevent ill-constructed hardware calls that may damage parts of the electrical and mechanical hardware. Higher level calibration tasks are also implemented, such as the calibration of the laser shutter and laser mirrors.

Eye_sim The simulator implementation forms a major software development phase of this project. It simulates most of the environmental functionality illustrated in figure 3.1. Even though development intensive, the simulator is an essential tool that allows controllers to be tested and trained before running them on the robotic hardware. This reduces learning time and damage to the robot, especially during initialisation of artificial intelligence (AI) controllers, such as neural networks and genetic algorithms.

4.4 Device Driver Level

The device drivers are exclusively used by the hardware control code and are specifically designed to interact with the robotic hardware. The implementation of this level was partially aided by the availability of third party open source products.

I^2C The I^2C protocol was originally developed by the company Philips to reduce the physical communication links between integrated circuits (ICs) and is used here to communicate between the computer and the P8000'2 control board. The protocol software that accompanied the P8000'2 was developed for single tasking MSDOS machines and had to be ported to LINUX as part of the software development phase of this research. Newer LINUX kernel releases support I^2C with a much higher timing accuracy. A P8000'2 board driver has also been implemented that runs with the upgraded kernels [122]. Compile time libraries [55] also exist that implement the I^2C protocol, although the timing accuracy has not been tested.

Bigphysarea This is a third party driver that controls the direct memory access (DMA) for image capture. This helps utilise hardware capabilities of the computer when capturing image sequences.

Meteor This is third party software that controls the operation of the image capture board.

4.5 Hardware Level

The hardware level is not part of the software, but is mentioned here to illustrate the interaction between the hardware and the software. Hardware specifications were given in chapter 3.

4.6 Simulator

The simulator is an important component of the experimental environment as it allows controllers to be tested and trained before they are run on the robotic hardware. Trials on a Pentium II 450

MHz with 256 MB RAM have shown that the simulator can reduce experimental trial times by a factor of 96, compared to the performance of the robot.

The simulator implementation was the most time consuming software development stage of this project, as it simulates most of the hardware device driver functionality that was developed for the hardware level. It also simulates mechanical properties that have not been investigated in this context before, in particular the use of solenoids for controlled camera positioning.

This section is not a mathematical excursion into vector and projection geometry, but it introduces some of the key issues involved in simulating the experimental environment.

4.6.1 The Projection Model

The projection model is the framework of the simulator and sets up the fundamental geometric representation. This model captures the projection dynamics, starting from the laser source to the projection of laser targets on the screen and the capture of target images by the camera of the mechanical MAVE.

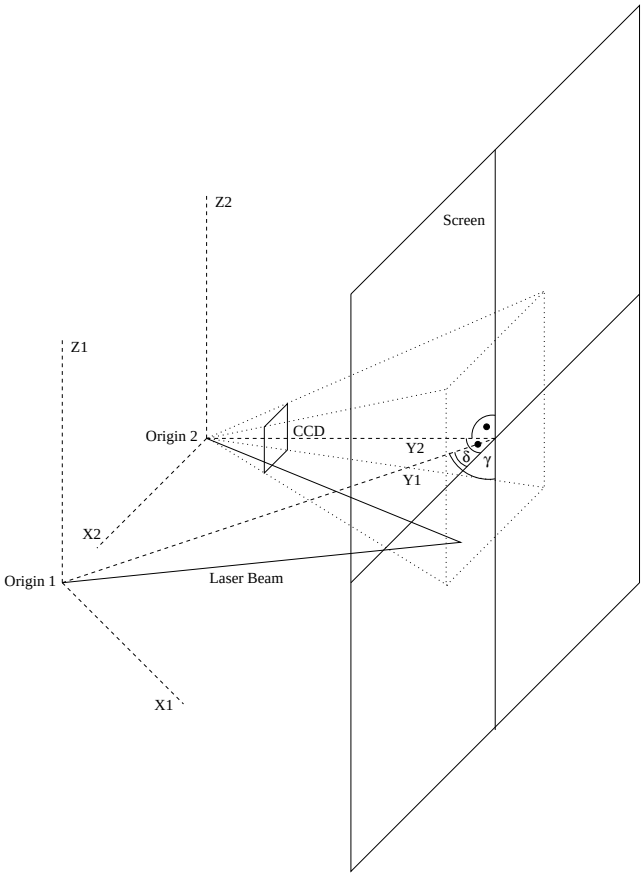


Figure 4.6: A simplified projection system, used to model the laser target position on the CCD array of the MAVE (scales, distance and distortions have been omitted here). *Origin 1* represents the origin of the laser beam and has the associated coordinate axes: *X1*, *Y1* and *Z1*. *Origin 2* represents the rotational centre of the gimbal, with the CCD array mounted slightly off centre. The associated coordinate axes to *Origin 2* are: *X2*, *Y2* and *Z2*. The screen forms the projection plane to which *Y1* has the horizontal and vertical angles γ and δ. *Y2* forms a right angle with the projection screen in both horizontal and vertical planes.

Figure 4.6 illustrates the simplified projection system that demonstrates the key features of the projection model. This projection system utilises two origins, *Origin 1* and *Origin 2* with the *Screen* as a common projection surface. The laser scanner in *Origin 1* has a variable coordinate system orientation to the projection *Screen*, defined by the angles γ and δ. This enables the simulator to be modified if the configuration of the physical environment changes at a later date. The MAVE, located in *Origin 2* is always assumed to be set perpendicular to the projection screen and has a fixed coordinate system, with respect to the projection screen.

This two origin approach enabled the laser scanner and MAVE properties to be modelled individually and combined later. In the first model *Laser Source to Screen Projection* information is projected from *Origin 1* to the *Screen*. In the second model *Screen to CCD Projection*, information is projected from the *Screen*, through the *CCD* array to *Origin 2*. The *CCD* array rotates around *Origin2* and simulates the movement of the camera in the mechanical MAVE.

The simulator code contains two high level procedure calls that implement these two projection models. The mathematical calculations that underlie these models can be found in [17, 72, 82, 96, 97, 123, 129] and are not covered here.

Laser Source to Screen Projection

The calculation of the laser source to screen projection is implemented to a fairly accurate degree, as the projection geometry from the laser source, figure 3.9, to the projection plane is well defined. The angular resolution of mirror rotations is also well defined, due to the use of stepper motors, see section 3.2. From a higher level, this means that, for given laser target control commands, it is possible to calculate the location of laser targets fairly well. Trials have shown that the simulation of the laser target system is accurate to within $\pm 0.825^{\circ}$ for horizontal movements and $\pm 0.5^{\circ}$ for vertical movements. This error calculation also takes account of backlash, generated by the reduction gears.

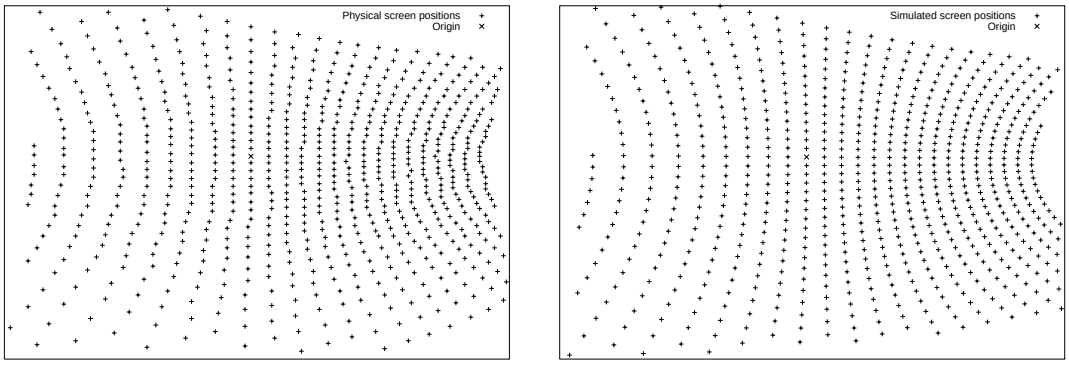


Figure 4.7: Laser target projection grid. The left image was produced by overlaying an accurate cartesian grid on the projection screen and measuring the location of laser target positions that occur when rotating the horizontal laser mirror in steps of 1.5° from one end of the screen to the other, then moving the vertical laser mirror by 1.5° and repeating the horizontal process. The right image represents a mathematical model of the measured positions.

Figure 4.7 compares the position of laser targets on the physical projection screen to targets on the simulated projection screen. The physical screen locations represent positions that were individually measured on an accurate cartesian grid, reducing inaccuracies that could occur by automatically measuring positions with a camera. The simulated locations are screen projections through the mathematical model.

Screen to Camera Projection

The projection of laser target positions from the screen to the CCD array are also geometrically well defined, as they follow a similar projection model as that implemented for the laser source to screen projections. The exception being that there is a projection from one plane in space to another plane, instead of a projection from an origin to a plane. In the mechanical environment, this projection is slightly distorted, due to the lens of the CCD camera. The distortions are negligible in the image centre but increase towards the image perimeter. Small projection corrections towards the image boundaries were implemented by mapping known real world locations to known image locations, captured by the CCD camera.

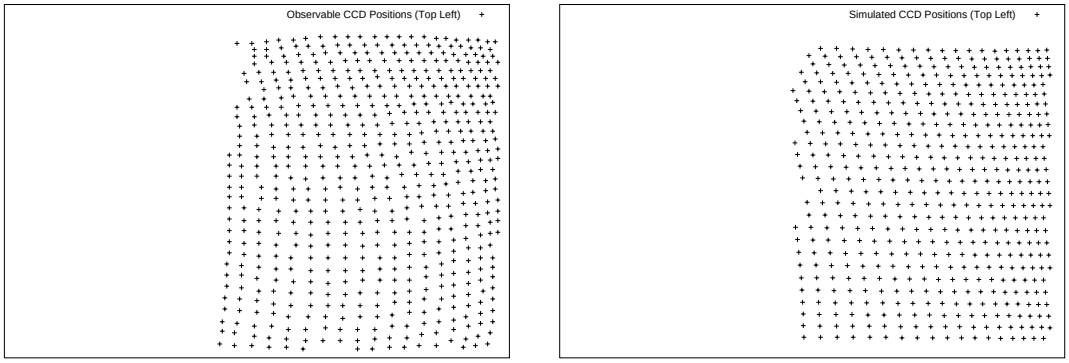


Figure 4.8: Laser target projection grid, viewed by the mechanical and simulated MAVE pointing at the top left corner of the screen. The left image was generated by the physical hardware and the right image was generated in simulation. The grids were generated by the same algorithms that were used to generate the target locations in figure 4.7, but with a positioning resolution of 0.75° instead of 1.5°. The physical target locations were measured automatically by image analysis, instead of manually measuring the positions. Distortion algorithms were also implemented for the simulation, to reflect the behaviour of the CCD lens.

Figures 4.8 and 4.9 show examples of the projection geometry, as observed by the stationary camera pointing in fixed directions towards the projection screen. Similar to figure 4.7, the laser scanner produces a grid of laser target locations on the projection screen. In these examples the inter target spacing is reduced, generating a higher density of laser targets. The images compare physical target locations on the left with simulated target locations on the right.

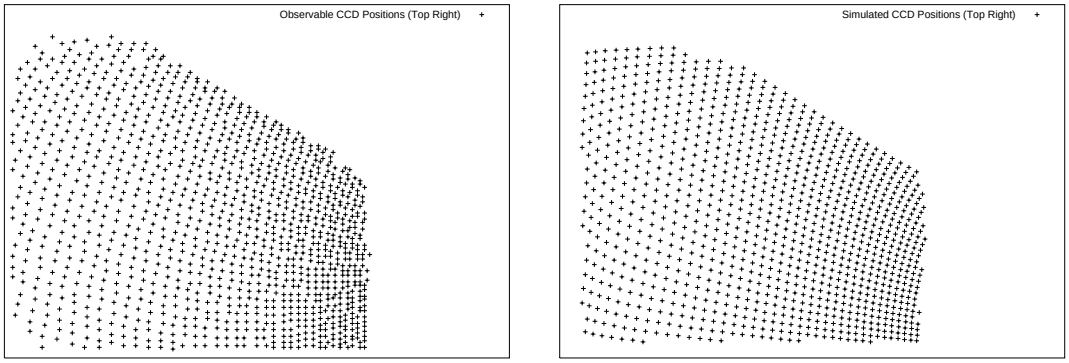


Figure 4.9: Laser target projection grid, viewed by the mechanical and simulated MAVE pointing at the top right corner of the screen. The left image was generated by the physical hardware and the right image was generated in simulation. The grids were generated by the same algorithms that were used to generate the target locations in figure 4.8. The physical target locations were measured automatically by image analysis. Distortion algorithms were also implemented for the simulation, to reflect the behaviour of the CCD lens.

4.6.2 Solenoid Activation

So far the simulator model is geometrically well defined and commands passed from the high level control produce similar results on the hardware and in simulation. The calculation of camera movement is an entirely different matter. As is shown in section 3.1, the camera is positioned by solenoids which apply non-linear forces. Contrary to the control of stepper motors, there is no defined positioning accuracy, and there is no literature that specifies the performance of solenoids in the configuration used here. It should be remembered that solenoids usually move between two physically defined positions.

The largest part of the simulator is taken up by simulating camera positioning as a result of solenoid activation. This is due to the complexity of solenoid control here and the mechanical effects that solenoid activations have on the movement of the gimbal. In the experimental environment, solenoids are controlled by activation time and pulse width (PW). The hardware is designed to run with 64 individual PW settings, allowing a wide range of activation signal combinations. Experiments have shown that only high activation times and PW values in the range 43 to 63 generate gimbal movements in all areas of the camera working envelope. PW values in the range 0 to 42 do generate camera movements but depend on the starting position of the camera in the working envelope. A second influence that affects the camera control is ambient temperature, see section 3.6. Solenoid activations in the lower PW range are influenced more strongly by ambient temperature than solenoid activations in the upper PW range. It was hence decided that the simulation should focus exclusively on the upper PW range, as this would simplify the simulator implementation.

The final relevant observation showed that the activation of solenoids produces two distinct camera motions. In this context, these motions or movements are considered to be of primary or secondary nature. Their definition is synonymous with that used in physiology, when describing

human eyeball positioning as a result of muscle contraction. Horizontal and vertical camera movements as a result of solenoid activation are referred to as primary or secondary, depending on the relative magnitude of each movement component. Primary movements are larger than secondary movements.

Primary motion / movement is primarily exhibited by the camera, if the direction of camera rotation and the direction of solenoid activation are the same. Primary motion may not occur if the activation parameters are too low and will not occur if a gimbal end stop is reached.

Secondary motion / movement is primarily exhibited by the camera, if the direction of camera rotation and the direction of solenoid activation are not the same. Secondary motion may not occur under certain alignments of the camera and will not occur if no primary motion is present.

For example: Primary camera movement occurs in the horizontal direction when generated by the activation of a solenoid that is responsible for horizontal activation. Secondary movement is the vertical movement component of the camera, when generated by the same solenoid. The presence of primary and secondary motion can be observed in both vertical and horizontal directions. Most, but not all primary movements are accompanied by secondary movement.

The implementation of the solenoid control and camera movement models relies on defining relationships between PW, activation time and the amount of primary and secondary camera motion. The extraction of parameters that allow these relationships to be established can be automated, but requires knowledge about the mechanical MAVE camera position. The measurement of camera rotation is implemented here by using the laser target as a reference point.

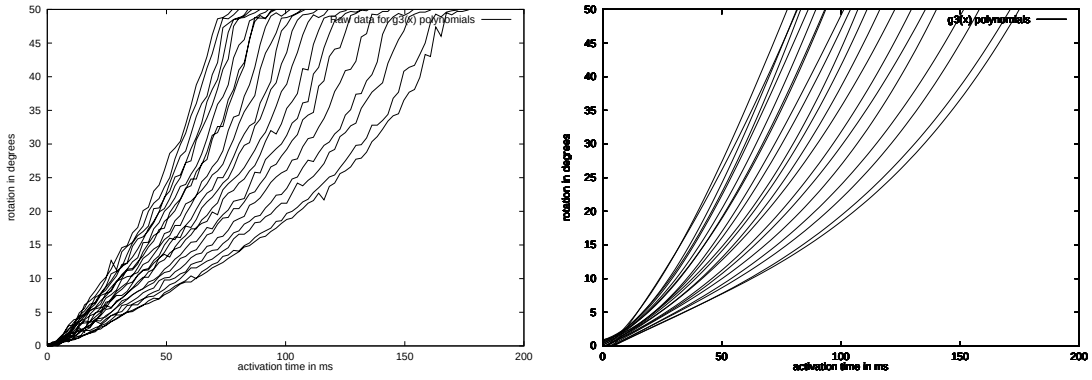


Figure 4.10: Plot of vertical gimbal rotation curves (top to bottom) for 21 PW settings. The solenoid activation time is given along the *X* axes, and the distance travelled is given along the *Y* axes. The left image was generated by repeatedly positioning the mechanical MAVE at the centre, top gimbal end position and activating the solenoid for vertical downward movement. Activation times started with *0ms* and were incremented by *1ms* for each following activation. The gimbal rotation was measured after each activation and corresponding graph locations were plotted. The right image shows a mathematical approximation of the collected data.

The first automated data acquisition technique generates a correlation model between PW, activation time and primary camera movement. The camera is positioned at an end stop, a solenoid

is then activated for a given amount of time and the amount of camera rotation is then measured. The process is repeated with increasing activation times for the whole upper PW range and all four solenoids. Figure 4.10 is a graphical representation of the data extracted and the corresponding mathematical models generated. The models allow the simulation of camera positioning for primary camera movements, starting from an end stop. As the solenoid activation forces are non-linear and depend on the starting position of the camera (extension of the solenoid plunger), a camera position dependent factor is introduced. This factor is taken directly from the manufacturer's literature, figure 3.13.

The tested performance of this first solenoid activation model is demonstrated for two directions in figure 4.11 and shows repetitive camera positioning for primary camera movements.

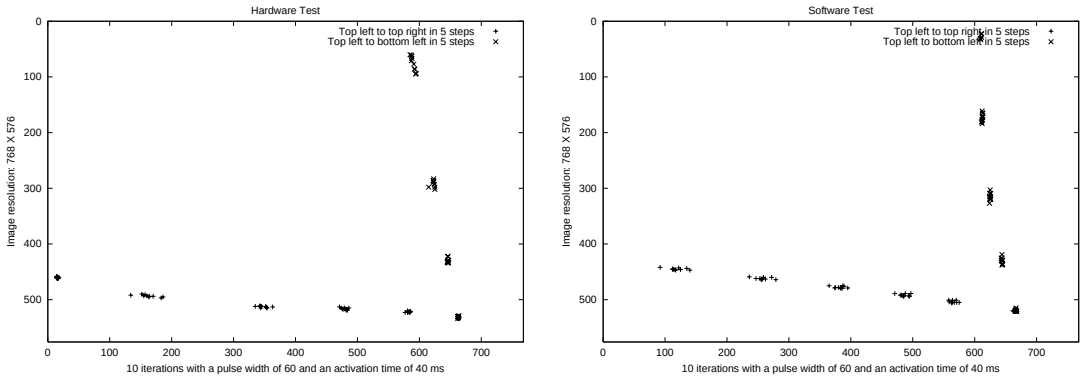


Figure 4.11: MAVE positioning repeatability tests. These images show a comparison of positioning accuracy during repetitive camera movements on the hardware and in simulation. The test environment was configured by pointing the mechanical MAVE straight ahead and positioning the laser target in the centre of the camera image array. This set up the laser target as a central reference point. After this basic configuration, two times five test batches were run: Before each batch the mechanical MAVE was moved to the extreme top left camera position. This placed the laser target into the bottom right corner of the image array. Batch 1: The solenoid for horizontal, left to right movement was activated five times for a duration of *40ms* and laser target positions in the image array were measured between each activation. Batch 2: The solenoid for vertical, top to bottom movements was activated five times for a duration of *40ms* and laser target positions in the image array were measured between each activation. The left image shows the results on the hardware and the right image shows the results in simulation.

The first model captures most of the camera control properties that are implemented in the simulator, but does not simulate secondary motion. Secondary motion occurs as a result of primary camera movement and is greatest towards the boundaries of the camera working envelope and smallest towards the centre of the working envelope. The amount of secondary movement depends on the starting position of the camera and the amount and direction of primary camera movement. The acceleration and speed of primary camera movement, controlled by PW and activation time, appear to play no noticeable role in generating secondary motion. For this reason it is possible to neglect the control values for PW and activation time in this second model.

The second automated data acquisition technique generates a correlation model between the camera starting position and primary and secondary camera motion. Similar to the extraction of the primary movement components, the laser target is again positioned as a reference point, allowing the extrapolation of the camera position from this point. The camera is then moved to extreme working envelope positions out of which maximum primary and secondary camera motion occurs. The camera is then continuously moved from top to bottom and bottom to top, or from left to right and right to left. The resulting secondary movement positions are extracted for all four solenoids. The correlation between measured, accumulated primary and secondary movement allows secondary motion to be calculated in simulation. Tests have shown that the measured relationship between primary and secondary movement holds for any location within the working envelope. Figure 4.12 is a graphical representation of the data extracted and the corresponding mathematical model generated.

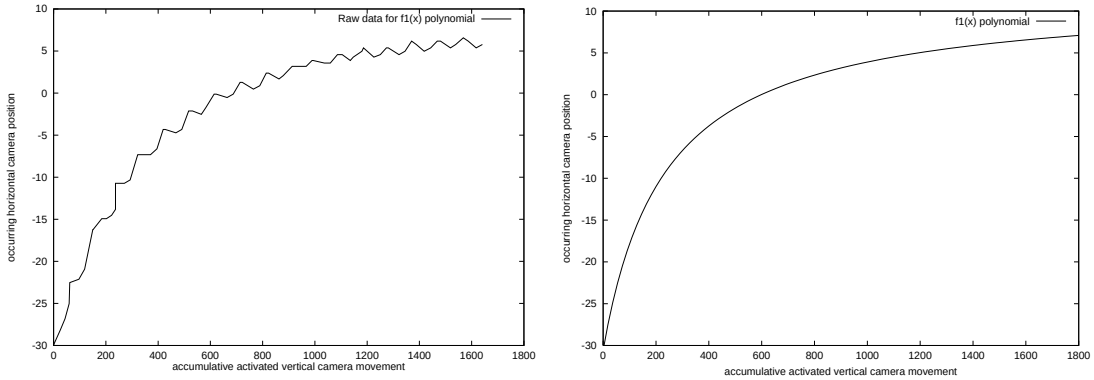


Figure 4.12: Plot of secondary camera movements. The test environment for the left image was configured by pointing the mechanical MAVE straight ahead and positioning the laser target into the centre of the camera image array. This set up the laser target as a central reference point at 0°. The mechanical MAVE was then moved to the extreme top right position. Vertical solenoid activations were then applied, moving the camera from top to bottom and bottom to top. The process was repeated until the accumulative vertical camera movement had reached 1600°. This primary vertical camera movement also produced a secondary horizontal camera movement, which started at -30° and progressed to 7°. The right image shows a mathematical approximation of the data collected. This data extraction and approximation process was also applied to measure the other three directions of secondary camera movement.

The performance of the second model is tested for two directions in figure 4.13. The camera is positioned at an extreme end stop from which camera activation produces a large amount of secondary movement. Repetitive horizontal camera movements are then applied and the secondary movement is measured.

After generating two models of mechanical camera movement properties, for primary and secondary movements, it is possible to simulate the combined control of camera movement. The simulation, similar to the hardware control, only requires information about PW and activation time. The current camera position in the working envelope is stored by two variables. The activation time, PW and camera position are used to calculate primary camera movement. The calculated primary movement and camera position are then used to calculate the secondary motion.

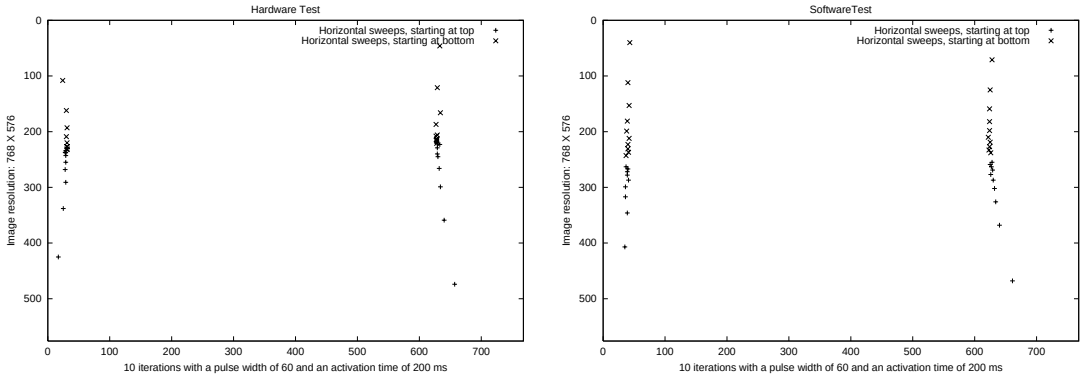


Figure 4.13: Mechanical MAVE positioning tests for secondary camera movement. The images show two tests for secondary vertical camera movement. The left image shows the results produced by the hardware. The right image shows the results produced in simulation. The test environment for the hardware was configured by pointing the mechanical MAVE straight ahead and positioning the laser target into the centre of the camera image array. This set up the laser target as a central reference point. After this configuration, the following two tests were run: 1: The mechanical MAVE was positioned at the extreme top left position and the horizontal actuators were alternately activated for ten times, moving the camera from the extreme left to the extreme right and back again. Position measurements were take each time an end position was reached. 2: The camera was then positioned at the extreme bottom right and the horizontal actuators were alternately activated for ten times, moving the mechanical MAVE from the extreme right to the extreme left and back again. Again position measurements were take each time an end position was reached. During these tests, the camera exhibited secondary vertical movements from the extreme top to the centre and respectively from the extreme bottom to the centre.

4.6.3 Simulator Performance

The results demonstrated in figures 4.11 and 4.13 only compare specific simulator properties with the hardware. These are very specific but restricted tests, as the laser target is not moved and the experimental cycles are very short. More realistic performance measures could be expected when comparing the results of complete experimental trials, run on the hardware and in simulation. This will inevitably reflect the performance of the camera controllers with regard to their target foveation or fixation capabilities. However, it can also be assumed that the exposure to the two operational environments will prompt the controllers to produce slightly different results. The following investigations look at these differences in more detail, but the controllers and their behaviour are not compared in detail. While observing the results of this evaluation, it should be kept in mind that the robotic hardware has unique positioning properties that are not position controlled and vary, strongly influenced by external influences, such as ambient temperature fluctuations.

In the previous chapter, the non-adaptive and adaptive saccadic controllers, to be further discussed in chapter 5 and analysed in chapter 6, helped evaluate the performance of the robotic hardware. The mathematical simplicity and close relationship between each of the controllers also makes them ideal tools for the evaluation of the simulator performance here. Tests are also run on the non-adaptive and adaptive smooth pursuit controllers. The log file parsing facility of the developmental environment allows simulator and hardware error graphs to be merged, providing

a good comparison method for the controllers run on each platform. Figures generated in this way form a central method for evaluating the simulator capabilities.

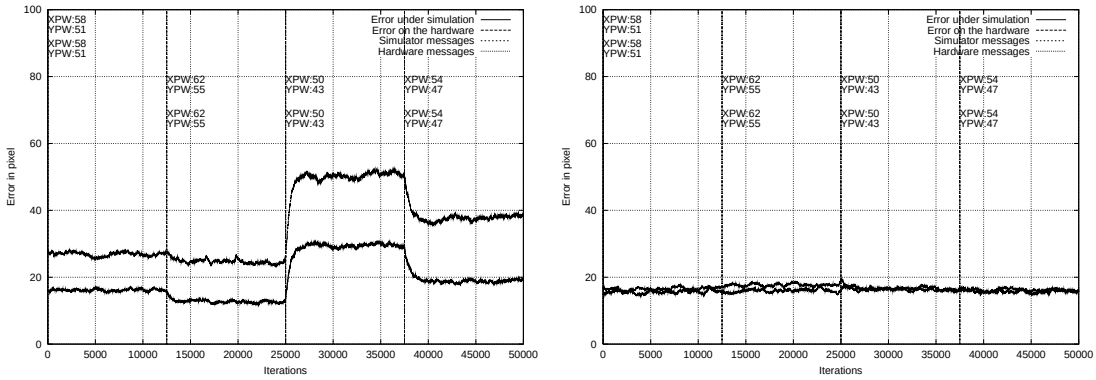


Figure 4.14: Simulator and hardware performance comparison for complete experimental runs of the two saccadic controllers. The graph on the left compares the simulator and hardware performance of the non-adaptive version. The graph on the right compares the simulator and hardware performance of the adaptive version. The four curves in the two graphs were smoothed by recursive averaging with a weight of 0.002.

Figure 4.14 compares the performance of the non-adaptive saccadic controller and the adaptive saccadic controller in simulation and on the hardware. The dashed vertical lines in each graph indicate test environment changes at which the PW values were altered. The laser scanner controller and MAVE controllers are not informed of these changes.

The performance of the non-adaptive controller in the left graph changes each time the PW is modified. The shape of the plot generated in simulation is similar to that generated on the hardware, but the relative error amplitude between them differs considerably. This suggests that the general model underlying PW processing appears to behave similarly in simulation and on the hardware. It also suggests that there is a continuous discrepancy between the simulator and the hardware. This discrepancy is hard to explain by only analysing the graph and could be a fundamental fault within the simulator or be due to minor inconsistencies between the simulator and the hardware. If this discrepancy is due to inconsistency, it may be assumed that this type of MAVE controller is very sensitive to these differences and demonstrates this in the error graph. In fact, it can be shown that a small reduction of the scaling constant s, (see section 5.2 for details) is sufficient to generate a simulation plot that is very close to the result generated on the hardware. The nearly indistinguishable error plots of the adaptive controller on the hardware and in simulation do not confirm, but can support this claim, as this controller adapts by modifying the value of s (see section 5.3 for details).

Figure 4.15 shows test runs of the non-adaptive smooth pursuit controller on the left and of the adaptive smooth pursuit controller on the right. The PW values were changed in the same way here, as was the case for the saccadic tests.

Again, the performance of the non-adaptive smooth pursuit controller changes each time the PW is altered, but the curves do not appear to be shifting in the same way as is the case with the saccadic controller. The curves exhibit a sinusoidal behaviour, which is due to the target pattern

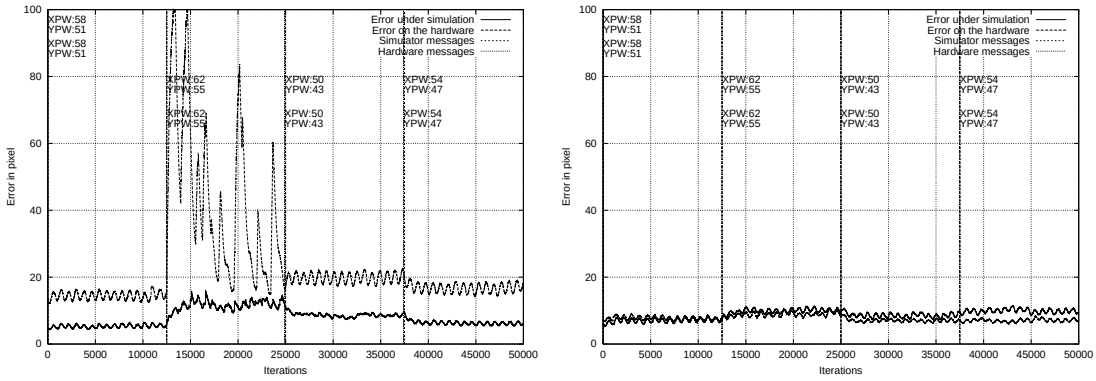


Figure 4.15: Simulator and hardware performance comparison for complete experimental runs of the two smooth pursuit controllers. The graph on the left compares the simulator and hardware performance of the non-adaptive version. The graph on the right compares the simulator and hardware performance of the adaptive version. The four curves in the two graphs were smoothed by recursive averaging with a weight of 0.002.

(see section 6.2 for more information), but the most extreme difference occurs during the trial phase with a PW setting of XPW:62 / YPW:55. At this point the hardware performance degrades significantly and error spikes overshoot the upper range of the scale. During this phase of the test, the mechanical MAVE falls into an oscillating behaviour, moving the camera from one side of the working envelope to the other. As the smooth pursuit controllers operate in a similar way as the saccadic controllers, it could again be assumed that a modification of s (see section 5.2 for details) would have an effect on the performance of the controller. Indeed, the adaptive smooth pursuit controller test can again confirm that s does play a considerable role in the performance of the controllers. The error graphs of the simulator and hardware are at times so similar that there appears to be hardly any difference between them. The adaptation during PW change is noticeable and expected. It shows again that the simulator, compared to the hardware, does appear to have similar properties, but the cause for the extreme performance difference of the non-adaptive controller still needs to be investigated. As is shown in the error graph and was apparent during the test, the camera oscillates and overshoots the target at certain PW values on the hardware. In the simulation, this does not appear to be the case. Figure 4.16 shows another representation of the left graph in figure 4.15. The curves generated by the simulator and by the hardware are plotted separately and are not smoothed. It becomes apparent that the average error of the simulator is lower than that of the hardware which is also evident in figure 4.15, but the simulator now also shows error peaks at XPW:62 / YPW:55. This indicates very inaccurate camera positioning and oscillation. The oscillation is not as strong in simulation as it is on the hardware, but as has been shown, the overall error performance is lower on the simulator than on the hardware, which could account for the fact that the oscillation is also lower.

Although these tests measure the performance of the simulator by analysing error graphs generated by controllers that run on the hardware and in simulation, the performance of the simulator most certainly can reflect general characteristics of the hardware. For the experiments run in chapter 6 this is more than sufficient, as the simulator primarily serves to protect the hardware from

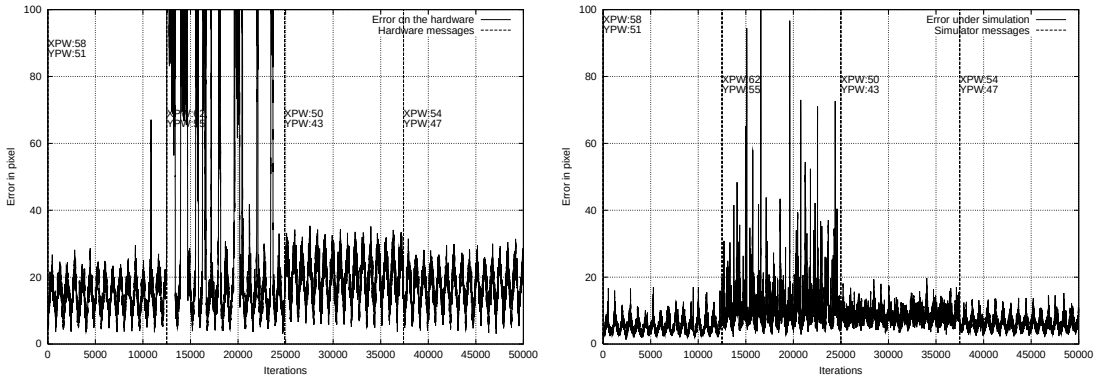


Figure 4.16: Simulator and hardware performance comparison for complete experimental runs of the non-adaptive smooth pursuit controller. The output in these graphs is not averaged, highlighting the error of every individual camera to target location. The graph on the left shows the controller performance on the hardware. The graph on the right shows the controller performance in simulation.

damage that may be caused by initial training periods of evolutionary or neural controllers. The simulator also provides a fast assessment system that helps assess the performance of prototype controllers.

4.7 Summary

The implementation of the software, outlined in this chapter, marks a major milestone of the project. It brings all areas of the research environment together. A general overview of the software hierarchy is first introduced, after which the individual components are covered in more detail. The in depth coverage of the simulator, in particular, conveys central issues that also aid the understanding of the control involved in operating the physical environment. Illustrations throughout the chapter compare data extracted from the hardware with data generated in simulation.

Chapter 5

Controllers

A range of experimental, low-level camera controllers are discussed in this chapter, which are essential for the movements of the Monocular Active Vision Eye (MAVE). These controllers use a simple target detection algorithm, which is part of the experimental environment and implements visual processing that returns no more than one unambiguous target location in an image grid. This type of target location representation avoids the need for selective gaze control algorithms. The low-level actuator controllers can utilise the image grid data directly in order to foveate[1] the camera on the target position. This type of low level camera control is not necessary in conventional position controlled active vision heads as they can be repeatedly positioned without requiring image feedback.

Two principal types of actuator controllers are discussed here: "saccadic" controllers, which enable ballistic camera movements to target locations and "smooth pursuit" controllers that enable tracking of targets. There are various control methods that enable the implementation of such controllers and the following sections introduce a combination of eight possible controllers and their main operational features.

5.1 Stationary Benchmark Controller

This controller is in a category of its own. Saccadic test patterns and smooth pursuit test patterns can legitimately be tested on this controller alike. The controller is the simplest model implemented here and is primarily used to measure the performance of target test sequences.

On engaging this controller, the camera is foveated on the initial target test pattern position, after which no more camera movements are performed. The initial target position of all target patterns in the experimental environment forms a central point around which all further target locations are distributed. Foveation on this point ensures that the camera is able to observe all following target locations that are part of the generated test pattern, without the need for camera repositioning.

[1] Foveate is used here to indicate active camera positioning that places an image of the target onto the centre of the CCD array.

5.2 Non-Adaptive Controllers

The non-adaptive saccadic and smooth pursuit controllers implement a closed loop, negative visual feedback mechanism, similar to that described by Young [133] p44. The central operation of both the saccadic and smooth pursuit systems are very similar to each other, except for the value of the scaling constant *s* and the actuator control rates. The common features of the controllers are introduced first.

The controllers measure the distance between the camera fovea and the target location and then try to minimize this *error* vector. The *error* is minimised by multiplying the horizontal and vertical vector components of the *error* with a scaling constant. The products are then passed to the activation unit that performs camera activation for the durations specified by the calculated products. Figure 5.1 shows a block diagram of the controller architecture that underlies both the saccadic and smooth pursuit control.

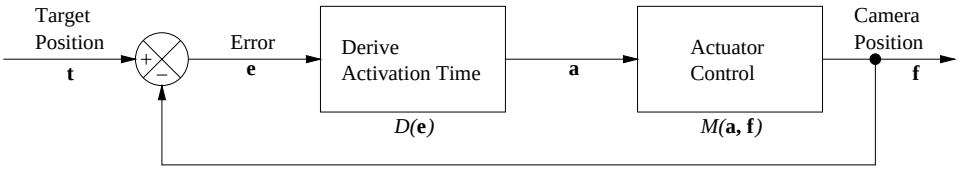


Figure 5.1: Linear, non-adaptive, servo mechanism for the low-level control of camera positioning.

The mathematical model for this controller architecture is as follows:

1. Measure the two dimensional *error* vector **e**, which lies on the image plane between the fovea **f** and the *target position* **t**. $|\mathbf{e}|$ is the *error* that is to be minimised:

$$\mathbf{e} = \mathbf{t} - \mathbf{f} \tag{5.1}$$

2. Calculate the activation vector **a** that defines the actuator activation times for horizontal and vertical movements. *s* is a scaling constant that is specific to the type of control that is to be performed by this controller, either smooth pursuit or saccadic movements:

$$\mathbf{a} = s\mathbf{e} \tag{5.2}$$

3. Engage the camera control with the activation vector **a** which generates a new *camera position* $\mathbf{f}'$:

$$\mathbf{f}' = M(\mathbf{a}, \mathbf{f}) \tag{5.3}$$

4. Go to step (1).

5.2.1 Saccadic

The saccadic controller is intermittently sampling and two control cycles are typically applied for each new target location. This means that after the target has been moved to a new location, the controller is permitted to perform two saccades in order to minimize the *error* between the target and camera fovea. This allows one large saccade to be followed by one smaller corrective saccade. This is in line with the saccadic control behaviour of the superior colliculus controller, discussed in section 5.5 and the physiological literature that confirms the presence of large saccades, followed by smaller corrective saccades [7, 11, 99].

The scaling constant s is set to 0.134. This value was derived by running the adaptive saccadic controller, discussed in section 5.3, for 10000 iterations with XPW:58 / YPW:51.

5.2.2 Smooth Pursuit

In nature, smooth pursuit control is believed to be continuously sampled, but the smooth pursuit controller described here is strictly speaking intermittently sampled. This is due to the configuration of the computing device on which the controller is run. However, experiments in the next chapter are set up to provide an environment that enables a very high sampling rate and small changes in adjacent target locations between each control cycle. In contrast to the saccadic control implementation, the smooth pursuit control allows not two but only one position update, each time the laser target is moved. This simulates a correction of the slip error that is produced by the pursuit controllers.

The scaling constant s is set to 0.251 and was derived in the same fashion as was the scaling constant for the saccadic controller, but here using the adaptive smooth pursuit controller. This value is high, compared to that of the saccadic controller and is due to the fact that the typical error during pursuit is much smaller than that during saccades. The activation time that controls the actuators is not proportional to the error or the distance that the camera has to travel. This is due to the fact that an initial period of activation time is spent on building up a sufficiently strong magnetic field, before any motion takes place. Further activation, beyond reaching this threshold, will result in movement of the camera. The scaling constant needs to take account of this behaviour. It can be assumed that this initial "start-up" time is, on average, the same for both smooth pursuit and saccadic movements. This is reflected in the high scaling constant for smooth pursuit. It ensures that activation times are large enough to overcome the "start-up" time and then still move the camera, even with very low target errors that are passed into the smooth pursuit controller.

5.3 Adaptive Controllers

The adaptive saccadic and smooth pursuit controllers are an extension of the controllers discussed under section 5.2 and the controller developed by Johnson and presented by Milhorn [79]. Similar to the controllers in the last section, the central operation of both the saccadic and smooth pursuit controllers are very similar to each other and their common features are discussed first.

In contrast to the last two controllers discussed, the adaptive controllers use s as a variable and not as a constant. Every time the camera is repositioned, the *error* is measured and used to adjust the value of the scaling factor s. The greater the *error*, the greater the adjustment to the scaling

factor *s*. *s* is increased if the fovea undershoots the target and is decreased if the fovea overshoots the target. Figure 5.2 shows a block diagram that outlines the operation of the adaptive saccadic and smooth pursuit controllers.

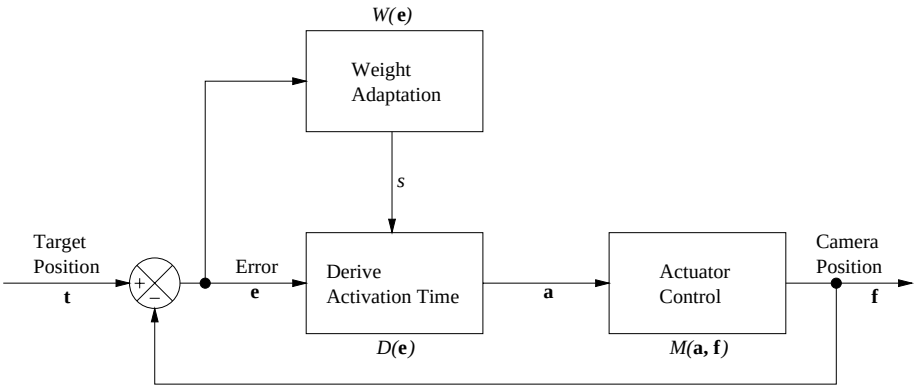


Figure 5.2: Linear, adaptive, servo mechanism for the low-level control of camera positioning.

The following mathematical model describes the operation of the adaptive control architecture:

1. Measure the two dimensional *error* vector **e**, which lies on the image plane between the fovea **f** and the *target position* **t**. $|\mathbf{e}|$ is the *error* that is to be minimised, see equation 5.1.

2. Calculate the activation vector **a** that defines the actuator activation times for horizontal and vertical movements. *s* is a scaling factor that is subject to change, see equation 5.2.

3. Engage the camera control with the activation vector **a**, which generates a new *camera position*, see equation 5.3.

4. Measure the two dimensional error vector $\mathbf{e}'$, which lies on the image plane between the fovea $\mathbf{f}'$ and the *target position* **t**. $\mathbf{e}'$ is the *error* that remains after the last camera movement. $\mathbf{e}'$ is used to evaluate the performance of the last camera movement:

$$\mathbf{e}' = \mathbf{t} - \mathbf{f}' \tag{5.4}$$

5. Calculate the *weight adaptation* for s' from the remaining *error* $\mathbf{e}'$. *v* defines the weight of the error $\mathbf{e}'$ in the calculation of s'. *p* is a negative penalty constant that is used when the fovea **f** overshoots the target **t**:

$$s' = \begin{cases} s + |\mathbf{e}'|vp & \text{if } \operatorname{acos}\left(\frac{\mathbf{e}.\mathbf{e}'}{|\mathbf{e}||\mathbf{e}'|}\right) > 90^\circ \quad \text{Target overshoot.} \\ s + |\mathbf{e}'|v & \text{if } \operatorname{acos}\left(\frac{\mathbf{e}.\mathbf{e}'}{|\mathbf{e}||\mathbf{e}'|}\right) \leq 90^\circ \quad \text{Target undershoot.} \end{cases} \tag{5.5}$$

6. Go to step (1).

5.3.1 Saccadic

The controller is intermittently sampled and applies two control cycles for each new target location, as is the case with the previously discussed saccadic controller. After each saccade is executed, the remaining *error* between the target and the fovea is used to change the weight of *s*.

The scaling factor s is initialised to 0.134 for saccades and is allowed to change as the controller adapts to the environment and other properties of the experimental set up. The penalty constant p is set to -10.0, which encourages target undershoot with the first saccade and closer target positioning with the following corrective saccade. v is set to 0.00001 and was experimentally determined, by initially setting v to 1 and reducing the value gradually until an equilibrium was reached. Larger values cause camera positioning to oscillate between target over- and undershoot. Smaller values do not cause oscillation but reduce the adaptation speed by which the controller can adjust to new experimental conditions. As the saccadic target pattern presents target locations with a random distribution, error values passed into the controller can fluctuate significantly. v is therefore set very low and prevents s from adapting quickly to large changes in $\mathbf{e}$. If the generated target locations were evenly distributed and required saccades of similar magnitude, v could be set to a larger value, allowing s to adapt at a faster rate.

5.3.2 Smooth Pursuit

The controller is sampled in a similar manner to the smooth pursuit controller previously discussed. The only difference being that the image is sampled twice per control cycle, instead of only once. This is necessary to evaluate the image slip after the camera position correction, which allows the scaling factor s to be adjusted.

The scaling factor s is initialised to 0.251 and is allowed to change as the controller adapts to the environment and other properties of the experimental set up. The parameter p is again set to the value -10.0, encouraging slip correction not to overshoot the target. This helps compensate for oscillation that can occur in the non-adaptive version of this controller. The value v is set to 0.0002 and was derived in the same way as was the case for the adaptive saccadic variant. Increases in this value show oscillations and decreases reduce the speed of convergence. In contrast to the saccadic target pattern, the pursuit target pattern produces target locations that are evenly spaced, but in random directions. This means that $\mathbf{e}$ does not change much from one target location to the next. s can therefore be allowed to adjust quicker to changes in $\mathbf{e}$, giving v a lower value.

5.4 Combined Least Squares Controller

The combined least squares controller is a logical extension to the adaptive saccadic and smooth pursuit controllers from the previous section. This controller has two parallel control paths, one implementing saccadic control, the other implementing smooth pursuit control. The switch to these paths operates by thresholding the absolute error between the fovea and the target location. If the error is above a certain value, the saccadic path is followed, otherwise the controller performs smooth pursuit. Once a control path has been selected, the camera remains in this mode until all associated control steps have been completed. This means, for example, that if the mode switch is set to saccadic control, the saccadic controller will perform up to three saccades in the attempt to foveate the target, even if the first saccade should prompt the switch to change to smooth pursuit.

The individual control paths are similar to those of the adaptive saccadic and smooth pursuit controllers. Both controllers try to minimise the error between the fovea and the target, either by saccades or by slip error correction. The performance of these camera movements is assessed by the difference in fovea and target error, before and after camera movements. There are two

fundamental changes to this controller, compared to the previous adaptive controllers. First, the saccadic path can execute between zero and three saccades, instead of exactly two, in order to foveate. This mechanism is biologically more plausible, as humans also apply a varying number of saccades to foveate on a target. Second, the scaling factor mechanism *s*, which assumes a factorised relationship between the camera activation time and camera to target error is replaced by the weights w_n (w_s and w_p for saccadic and pursuit respectively). These weights are variable factors that are used to calculate activation times and are polynomial results of the fovea to target error. This means that the activation times are not only the product of the fovea to target error and an activation factor, but are a polynomial result of the fovea to target error. As a consequence, the activation weights are far more capable of adapting to the physical properties of the Monocular Active Vision Eye (MAVE) and behaviours of the laser test patterns. Although this method can adapt better to changes in target behaviour over the entire visual range, the controller requires more time to adapt to changes in the physical camera dynamics. This is due to the fact that more information is required to invoke changes in the saccadic and pursuit functions.

The following diagram shows the logical structure of the controller with the dual control path:

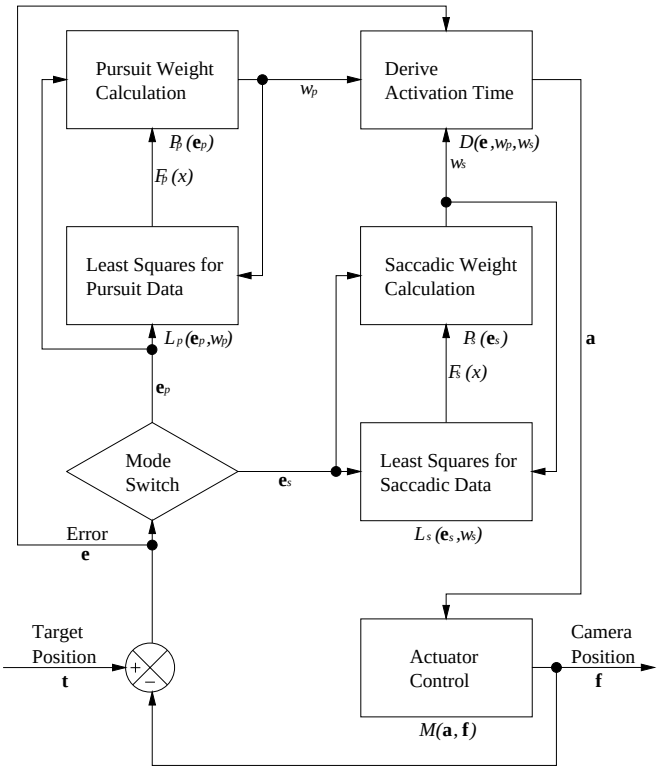


Figure 5.3: Dual path adaptive feedback mechanism for the combined control of smooth pursuit and saccadic camera control.

The mathematical model describes the operation of the adaptive dual path control architecture. In this context, *n* is used to refer to the saccadic or pursuit path, which in figure 5.3 are symbolised by *s* and *p* respectively:

1. Measure the two dimensional *error* vector **e**, which lies on the image plane between the fovea **f** and the *target position* **t**. $|\mathbf{e}|$ is the *error* that is to be minimised, see equation 5.1.

2. Decide whether $|\mathbf{e}|$ should be minimised by saccadic or smooth pursuit control, with $\mathbf{e}_n$ being the error provided to the saccadic or pursuit path:

$$\mathbf{e}_n = \begin{cases} \mathbf{e}_s & \text{if } |\mathbf{e}| > b \quad \text{Saccadic Path.} \\ \mathbf{e}_p & \text{if } |\mathbf{e}| \leq b \quad \text{Pursuit Path.} \end{cases} \tag{5.6}$$

3. Calculate the activation weight w_n, which is a function of the absolute error $|\mathbf{e}_n|$:

$$F_n(|\mathbf{e}_n|) = w_n \tag{5.7}$$

4. Derive the activation vector $\mathbf{a}$ that defines the activation times for horizontal and vertical camera movement. This calculation performs the same operation as that found in equation 5.2, although the scaling value w_n is derived differently:

$$\mathbf{a} = w_n \mathbf{e} \tag{5.8}$$

5. Engage the camera control with the activation vector $\mathbf{a}$, which generates a new *camera position*, see equation 5.3.

6. Measure the two dimensional error vector $\mathbf{e}'$, which lies on the image plane between the fovea $\mathbf{f}'$ and the *target position* $\mathbf{t}$. $\mathbf{e}'$ is the *error* that remains after the last camera movement. See equation 5.4 for details.

7. Bypass the *mode switch* and follow the path already set:

$$\mathbf{e}'_n = \begin{cases} \mathbf{e}'_s & \text{if } |\mathbf{e}| > b \quad \text{Saccadic Path.} \\ \mathbf{e}'_p & \text{if } |\mathbf{e}| \leq b \quad \text{Pursuit Path.} \end{cases} \tag{5.9}$$

8. Derive an activation weight w'_n for the error $|\mathbf{e}_n|$ to form a data pair $(|\mathbf{e}_n|, w'_n)_n$ that is added to the looped data pool for least squares calculations. w'_n should be an improvement to w_n for the argument $|\mathbf{e}_n|$:

$$w'_n = \begin{cases} F_n(|\mathbf{e}_n|) + F_n(|\mathbf{e}'_n|)vp & \text{if } \operatorname{acos}\left(\frac{\mathbf{e}_n \cdot \mathbf{e}'_n}{|\mathbf{e}_n||\mathbf{e}'_n|}\right) > 90^\circ \quad \text{Target overshoot.} \\ F_n(|\mathbf{e}_n|) + F_n(|\mathbf{e}'_n|)v & \text{if } \operatorname{acos}\left(\frac{\mathbf{e}_n \cdot \mathbf{e}'_n}{|\mathbf{e}_n||\mathbf{e}'_n|}\right) \leq 90^\circ \quad \text{Target undershoot.} \end{cases} \tag{5.10}$$

9. Derive the coefficients c_0 to c_m of the m degree polynomials $F_n(x)$ by calculating the least squares approximation to the pool of data pairs $(x_i, y_i)_n$. Note that an algorithm optimisation allows each matrix location to be computed by no more than two multiplications, two factorised subtractions and two factorised additions:

$$\begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_m \end{bmatrix} = \begin{bmatrix} \sum_{i=0} x_i{}^0 & \sum_{i=0} x_i{}^1 & \dots & \sum_{i=0} x_i{}^m \\ \sum_{i=0} x_i{}^1 & \sum_{i=0} x_i{}^2 & \dots & \sum_{i=0} x_i{}^{m+1} \\ & & \vdots & \\ \sum_{i=0} x_i{}^m & \sum_{i=0} x_i{}^{m+1} & \dots & \sum_{i=0} x_i{}^{2m} \end{bmatrix}^{-1} \times \begin{bmatrix} \sum_{i=0} x_i{}^0 y_i \\ \sum_{i=0} x_i{}^1 y_i \\ \vdots \\ \sum_{i=0} x_i{}^m y_i \end{bmatrix} \tag{5.11}$$

10. Generate new weight calculation functions $F'_n(x)$ with the coefficients c_0 to c_m:

$$F'_n(x) = \sum_{j=0}^{m} c_j x^j \tag{5.12}$$

11. Go to step (1).

An outer loop controls for how many iterations each of the two paths can run, before returning control to the experimental environment. If in smooth pursuit mode, control is given back each time one cycle of the loop has been executed. If in saccadic mode, control is given back to the environment after no more than three cycles or as soon as the fovea is within five pixels of the target. Other values and constants used in this controller are defined as follows:

- b is the switch threshold which determines the control path within the controller, either saccadic or smooth pursuit. This value is set to 40 pixels and was derived by approximating the maximum error produced by the smooth pursuit target controller.
- i represents entities of $(x,y)_s$ and $(x,y)_p$ data pairs that are used to derive the polynomials $F_s(x)$ and $F_p(x)$. There are typically 450 data pairs for each of the two polynomial calculations. There is no fixed rule by which this value was derived, but smaller values result in faster data approximations and large fluctuations between consecutively derived polynomials. Larger values allow a smooth progression from one derived polynomial to the next, but adaptations are slow.
- m specifies the degree of either the saccadic and smooth pursuit polynomials $F_n(x)$. $F_s(x)$ is set to 2 and $F_p(x)$ is set to 4, which describes polynomials of the following form: $F_s(x) = c_2x^2 + c_1x + c_0$ and $F_p(x) = c_4x^4 + c_3x^3 + c_2x^2 + c_1x + c_0$. These degrees of polynomials were determined experimentally and were found to produce the best results. Polynomials of lower degree produced worse positioning results and polynomials of higher degree produced no noticeable improvement.
- p specifies the negative penalty that is applied to camera activations when targets are overshot. This is set to -10 and is identical to the value used for the adaptive smooth pursuit and saccadic controllers.
- v is the weight with which modifications are applied to the second value of data pairs that enter the data pool. The value is set to 0.06 for both saccadic and pursuit calculations. This value is directly linked to i. Both the least squares calculation, using i and the weighted average, using v are approximation techniques that influence the behaviour of $F_p(x)$ and $F_s(x)$. During the calibration of i, v also had to be modified. Large values for i and small values for v cause small changes in $F_p(x)$ and $F_s(x)$. Small values for i and large values for v cause large changes in $F_p(x)$ and $F_s(x)$, which also leads to camera oscillations.

Although this controller adapts more slowly to changes in the mechanical properties than the previously discussed adaptive controllers, it forms a very important first step away from the experimental environment and towards the implementation of a gaze control algorithm that can interact with the natural environment. In nature targets behave unpredictably and eye control needs to be able to switch between smooth pursuit and saccadic movements.

5.5 Dog Net Controller

The *dog net* controller used here was previously implemented by Christopher Burdess [24], following the implemented model of the superior colliculus by Hege Ritter et al [98] and can learn saccadic motor control. This controller has, until now, not been tested in hardware, due to the lack of adequate hardware systems. A related model [12] does, however, exist that has been implemented on a conventional active vision head, but it requires the conversion of activation weights into a range of motor control signals. The initial work behind all of these controllers is based on the foveation hypothesis by D. A. Robinson [103], which was discussed in more detail in subsection 2.2.1.

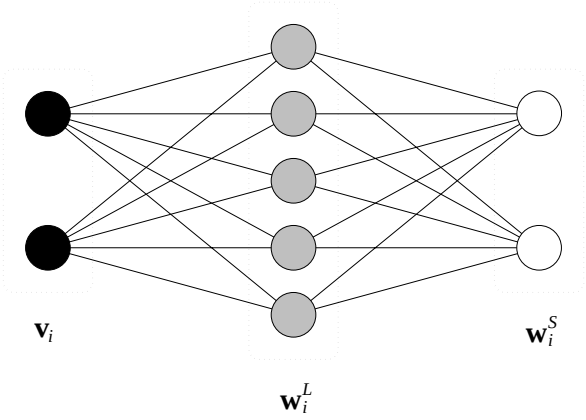


Figure 5.4: Simplified schematic of the dog net architecture. Black nodes represent the location of stimulation ($\mathbf{v}_i$), grey nodes represent the lattice units ($\mathbf{w}_i^L$) and white nodes represent the direction and length of saccades ($\mathbf{w}_i^S$). The value *i* is the node index and lies in the range 0 to 599.

This neural controller implements a topologically conserving feature map with one layer of input nodes and two layers of neurons that simulate the behaviour of the superior colliculus. Figure 5.4 shows a simplified schematic representation of the dog net structure. The original network and the one implemented here use 600 nodes in each layer, arranged in two sets of 20 concentric rings with 30 nodes each. One layer represents the location of stimulation in the field of view (black nodes), the second layer represents the location of the lattice units (gray nodes) and the third layer represents activation weights that are associated with the locations of the lattice nodes (white nodes).

In the original implementation, the direction and length of saccades was represented by vectors ($\mathbf{w}_i^S$) that pass from lattice node locations ($\mathbf{w}_i^L$) to the centre of the network. This is represented graphically in the left image of figure 5.5. It is clearly visible that the saccadic weights from each lattice node location do form a vector that points nearly perfectly into the centre of the network. While porting this dog net to the developmental environment, it was not possible to retain this representation of activation weights. The activation weights were replaced by activation times which are required by the physical and simulated solenoids to move the camera into the fovea. However it is still possible to display these weights in a graphical fashion. This is shown in the right image of figure 5.5.

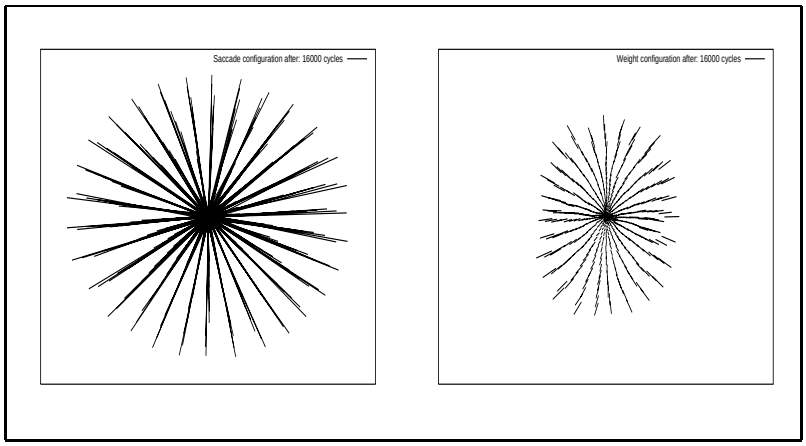


Figure 5.5: Graphical representation of the saccadic weights ($\mathbf{w}^S$) for the dog net. The image on the left shows the weight representation as is used for the original dog net. In this case the length and direction of the weights represent lines from network nodes to the centre of the network. The image on the right shows the weight representation as is used for the mechanical MAVE. The length and the direction of the weights represent activation times for the camera actuators.

The lattice locations of the dog net can also be displayed in a graphical fashion. This is shown in the left image of figure 5.6. The target pattern of the original dog net produces target locations with a Gaussian distribution, placing the lattice nodes in a fairly even distribution. The lattice nodes of the ported dog net are shown in the right image of figure 5.6. It is directly apparent that the distribution of the lattice nodes is very much distorted, compared to the locations of the original dog net. This is purely a result of the target pattern used on the MAVE. It was necessary to compress the Gaussian distribution of target locations horizontally to prevent the camera from hitting physical end positions.

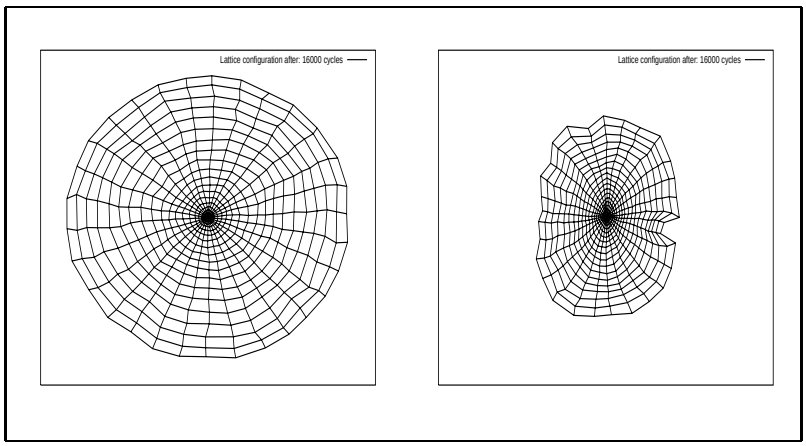


Figure 5.6: Graphical representation of the lattice node locations ($\mathbf{w}^L$) for the dog net. The image on the left shows the locations generated by the original dog net. The symmetric distribution of the nodes is generated by target locations that have a Gaussian distribution. The image on the right shows the lattice node locations as is generated by the mechanical MAVE.

The graphical representation of the original and ported dog net controllers in figures 5.5 and 5.6 show the weight topology after 16000 iterations. After this number of iterations, the lattice locations and the saccadic / activation weights have become fixed and the network stops adapting. As small test sequences in the previous chapters have shown, the activation behaviour of the MAVE does not follow a linear pattern, and activation times can differ considerably, depending on a range of external influences. This leads to the assumption that the performance of the dog net may vary on the MAVE, depending on the degree of convergence and influences from the experimental environment.

The algorithm that implements the model of the superior colliculus operates as follows:

1. Present a target vector $\mathbf{v}^L$ which lies between the fovea **f** and the target **t**.

$$\mathbf{v}^L = \mathbf{t} - \mathbf{f} \tag{5.13}$$

2. Determine the centre of excitation $\mathbf{c}^L$ in the lattice $\mathbf{w}_i^L$, by minimising $|\mathbf{v}^L - \mathbf{w}_\Delta^L|$ through the following condition:

$$\mathbf{c}^L = \mathbf{w}_\Delta^L \quad \text{for} \quad |\mathbf{v}^L - \mathbf{w}_\Delta^L| \leq |\mathbf{v}^L - \mathbf{w}_i^L| \forall i \tag{5.14}$$

3. Perform the learning step to form a Kohonen-like topology-conserving map onto the lattice. The Gaussian neighbourhood function $h^L_{(\mathbf{c}^L, i, \sigma^L)}$ (equation 5.22) and the learning rate decay with σ^L and η^L respectively (equation 5.23):

$$\Delta \mathbf{w}_i^L = \mathbf{w}_i^L + \eta^L h^L_{(\mathbf{c}^L, i, \sigma^L)} (\mathbf{v}^L - \mathbf{w}_i^L) \forall i \tag{5.15}$$

4. Calculate the activation vector $\mathbf{a}^S$ centred on $\mathbf{c}^L$, by summing and averaging the Gaussian distributed activation weights. Then trigger the saccade:

$$\mathbf{a}^S = \frac{\sum_i h^S_{(\mathbf{c}^L, i, \sigma^L)} \mathbf{w}_i^S}{\sum_i h^S_{(\mathbf{c}^L, i, \sigma^L)}} \tag{5.16}$$

5. Measure the new target vector $\mathbf{v}^S$ which lies between the fovea position $\mathbf{f}'$ and the target location **t**:

$$\mathbf{v}^S = \mathbf{t} - \mathbf{f}' \tag{5.17}$$

6. If the image is now in the fovea, i.e. $|\mathbf{v}^S| < r_{fovea}$, go to step (1).

7. Find the new centre of excitation $\mathbf{c}^S$ in the lattice $\mathbf{w}_i^L$, by minimising $|\mathbf{v}^S - \mathbf{w}_\Delta^L|$ through the following condition:

$$\mathbf{c}^S = \mathbf{w}_\Delta^L \quad \text{for} \quad |\mathbf{v}^S - \mathbf{w}_\Delta^L| \leq |\mathbf{v}^S - \mathbf{w}_i^L| \forall i \tag{5.18}$$

8. Calculate the activation vector $\mathbf{a}^{S'}$ on $\mathbf{c}^S$, by summing and averaging the Gaussian distributed activation weights. Then trigger the second corrective saccade:

$$\mathbf{a}^{S'} = \frac{\sum_i h^S_{(\mathbf{c}^S, i, \sigma^S)} \mathbf{w}_i^S}{\sum_i h^S_{(\mathbf{c}^S, i, \sigma^S)}} \tag{5.19}$$

9. Measure the new target vector $\mathbf{v}^{S'}$ which lies between the fovea position $\mathbf{f}''$ and the target location $\mathbf{t}$:
$$\mathbf{v}^{S'} = \mathbf{t} - \mathbf{f}'' \tag{5.20}$$

10. If this corrective saccade is an improvement, i.e. $|\mathbf{v}^{S'}| < |\mathbf{v}^{S}|$, perform the saccadic weight learning step. The Gaussian neighbourhood function $h^{S}_{(\mathbf{c}^{L},i,\sigma^{S})}$ (equation 5.22) and the learning rate decay with σ^{S} and η^{S} respectively (equation 5.23):
$$\Delta\mathbf{w}_i^{S} = \mathbf{w}_i^{S} + \eta^{S} h^{S}_{(\mathbf{c}^{L},i,\sigma^{S})}(\mathbf{a}^{S} + \mathbf{a}^{S'} - \mathbf{w}_i^{S}) \forall i \tag{5.21}$$

11. Go to step (1).

The terms $h^{L}_{(u,i,\sigma)}$ and $h^{S}_{(u,i,\sigma)}$ are simple Gaussian functions of the distance $|i-u|$ between circumferential and radial network locations c and r, depending on σ^{L} and σ^{S}, which decrease as the network converges:

$$h_{(u,i,\sigma)} = \exp\left(-\frac{\sqrt{\left(\left(r_i sin\left(\frac{2\pi c_i}{C}\right)\right) - \left(r_u sin\left(\frac{2\pi c_u}{C}\right)\right)\right)^2 + \left(\left(r_i cos\left(\frac{2\pi c_i}{C}\right)\right) - \left(r_u cos\left(\frac{2\pi c_u}{C}\right)\right)\right)^2}}{2\sigma}\right) \tag{5.22}$$

The learning rates η^{L} and η^{S} also decrease as the network converges, according to standard exponential decay, with t indexing the network learning cycles. The parameters used here are identical to those of the original dog net implementation. Burdess [24] does not specify how these values were derived, but this type of network is very robust and other parameter combinations may also be possible.

> "... the neighbourhood size is reduced with time during the training sequence. But how quickly do we reduce it and to what final size? Unfortunately there are no hard and fast rules for adaptive training algorithms of this nature and some experimentation will be required in individual applications. However Kohonen does stress that his method is not one that is brittle – that is, small changes in system parameters do not reflect gross divergence of training results... ." R. Beale and T. Jackson [13], p119

$$\begin{aligned}
\eta^{L}(t) &= 0.3 \exp\left(-0.0002t\right) \\
\eta^{S}(t) &= 0.3 \exp\left(-0.0001t\right) \\
\sigma^{L}(t) &= 10 \exp\left(-0.0003t\right) \\
\sigma^{S}(t) &= 3 \exp\left(-0.0003t\right)
\end{aligned} \tag{5.23}$$

As can be seen in the saccadic weight learning rule from equation 5.21, a saccadic weight adaptation is made each time two consecutive saccades produce an improvement on target foveation. There is no mechanism that specifies what type of action should be taken if a saccade was to overshoot the target or produce a worse result. This means that in a non-linear environment, such as that of the experimental environment, it can be expected that the controller will produce undesirable results, possibly by constantly overshooting or undershooting the target.

5.6 Adaptive Dog Net Controller

As the camera control and the experimental environment are dynamic and subject to change, it can be assumed that an adaptive neural network would be better at responding to changes than a converging one. The following network consists of a converging and a non-converging layer of neurons and is derived from the previously discussed dog net model. In this version, the lattice node positions of the network are allowed to converge, just like the previous model. This has two reasons: First, the lattice node positions are not heavily influenced by the controlled changes of the experimental environment. Second, the saccadic activation weights are relative to the location of the lattice nodes. This means that if the lattice nodes change their location, the saccadic activation weights would also have to be changed accordingly.

The saccadic weights can either converge to fixed values, or change in response to the dynamics of the experimental environment. This operation is controlled by altering the decay of η^S and σ^S. They do not converge as a function of t. In this version, equation 5.23 is replaced by equation 5.24, which allows the saccadic activation weights to decreased as well as increase as a function of e.

$$\begin{aligned} \eta^L(t) &= 0.3 \exp\left(-0.0002t\right) \\ \eta^S(e) &= 0.3 \exp\left(-0.0001e\right) \\ \sigma^L(t) &= 10 \exp\left(-0.0003t\right) \\ \sigma^S(e) &= 3 \exp\left(-0.0003e\right) \end{aligned} \tag{5.24}$$

e is steadily incremented with t if the fovea does not overshoot the targets. As soon as target locations start to be overshot, e is decremented, reversing the convergence of the saccadic weight adaptation.

$$e' = \begin{cases} e-1 & \text{if } \operatorname{acos}\left(\frac{\mathbf{v}^S.\mathbf{v}^{S'}}{|\mathbf{v}^S||\mathbf{v}^{S'}|}\right) > 90^\circ \quad \text{Target overshoot.} \\ e+1 & \text{if } \operatorname{acos}\left(\frac{\mathbf{v}^S.\mathbf{v}^{S'}}{|\mathbf{v}^S||\mathbf{v}^{S'}|}\right) \leq 90^\circ \quad \text{Target undershoot.} \end{cases} \tag{5.25}$$

The changes in this controller do not influence the learning rule of the dog net, that combines the weights of two consecutive saccades, if they are an improvement to the first saccade. Similarly to the original dog net, it can be expected that this controller will also allow target overshoot and a low performance to the non-linearity of the environment. The overall performance of the network should however be an improvement on the previous one, especially when confronted with environment and PW changes.

5.7 Summary

This chapter introduces camera controllers that enable the MAVE to pursue and saccade to targets, which are generated in a controlled environment. The design of these controllers is based on existing theories about the low-level control of human eye movements that could so far only be tested in simulation. These cover in particular control theoretical and neural controllers with adaptive and non-adaptive variants. Mathematical and diagrammatic representations help convey

the operation of these controllers. Although the controllers only form a small selection of possible designs, they do demonstrate a controller level in active vision systems that is generally not required in conventional platforms.

Chapter 6

Results

The results presented here cover the performance of the mechanical Active Vision Eye (MAVE) and the software controllers that perform smooth pursuit and saccadic camera movements. As hardware measurements and calibrations have already been covered in chapter 3, the results here only contrast the performance of the mechanical MAVE with the performance of other selected active vision platforms. The MAVE controller tests cover by far the largest part of this chapter and their performance is measured in the controlled environment, introduced in detail in chapters 3 and 4. This environment provides unambiguous and precisely definable target locations that can be detected within the camera's field of view. The camera controllers tested here utilise this target location information to implement camera positioning.

In the following tests, the positioning objective of all controllers is to minimise the distance between the fovea and the laser target location. The quality and reliability of this positioning control is graphically illustrated in *error graphs*. These graphs provide a relationship between the number of test iterations and the Pythagorean distance in pixels, which remains between the fovea and the target, after a controller has attempted to foveate the target. The error graphs also provide information on the value of pulse width (PW) settings which are used to apply controlled changes to the behavioural properties of the camera. These changes are used here to simulate behaviours that are similar to fatigue in the human extra-oculomotor system, to which a controller may be able to adapt. The laser target and camera controllers are not informed of these PW values and possible changes to them. Depending on the adaptability and performance of the camera controllers, these PW changes can have a significant impact on the foveation behaviour of the MAVE, which is reflected in the error graphs.

6.1 Hardware Performance

Table 2.1 in chapter 2 compares the performance of five different active vision platforms. These platforms were selected on the merits of their design diversity. Table 6.1 lists the same active vision platforms again, but this time also includes the measured properties of the mechanical MAVE. As all of these platforms have a different mechanical architecture, it is not easy to compare them by their individual physical properties. Therefore the platforms are compared by their performance

	Parallel Architectures				Serial Architectures	
	Monocular Vision			Stereo Vision		
	MAVE	HPV [21]	Agile Eye [47]	CeDAR [116]	ESCHeR [61]	Yorick [112]
Max Pan Acc	$16900^\circ/s^2$	$72000^\circ/s^2$	$20000^\circ/s^2$	$20000^\circ/s^2$	$4000^\circ/s^2$	$3000^\circ/s^2$
Max Pan Vel	$930^\circ/s$	$600^\circ/s$	$1000^\circ/s$	$800^\circ/s$	$140^\circ/s$	$150^\circ/s$
Max Pan Ang	$60.5^\circ \pm 5.4^\circ$	120°	$140^\circ \pm 30^\circ$	90°	200°	360°
Pan Res	$< 0.068^\circ$	0.01°	$0.135^\circ \pm 0.045^\circ$	0.01°	0.0044°	0.00018°
Max Tilt Acc	$17600^\circ/s^2$	$72000^\circ/s^2$	$20000^\circ/s^2$	$18000^\circ/s^2$	$14000^\circ/s^2$	$5000^\circ/s^2$
Max Tilt Vel	$913^\circ/s$	$600^\circ/s$	$1000^\circ/s$	$600^\circ/s$	$350^\circ/s$	$400^\circ/s$
Max Tilt Ang	$61.7^\circ \pm 15.3^\circ$	180°	$140^\circ \pm 30^\circ$	90°	90°	360°
Tilt Res	$< 0.068^\circ$	0.01°	$0.135^\circ \pm 0.045^\circ$	0.01°	0.0145°	0.00036°
Max Verg Acc	n/a	n/a	n/a	n/a	$16000^\circ/s^2$	$6000^\circ/s^2$
Max Verg Vel	n/a	n/a	n/a	n/a	$400^\circ/s$	$400^\circ/s$
Max Verg Ang	n/a	n/a	n/a	n/a	100°	360°
Verg Res	n/a	n/a	n/a	n/a	0.0125°	0.00036°

Table 6.1: A comparison of hardware performances characteristics for six different active vision platforms, including the mechanical MAVE developed as part of this research. This table contains a selection of architectures that can be found in most active vision systems currently available.

results listed in the table.

With a pan acceleration of $16900^\circ/s^2$, the mechanical MAVE lies in the mid range of the other five platforms, except for HPV, which outperforms all other platforms by far. It is also interesting to note that the pan acceleration of the parallel architectures is higher than that of the serial architectures listed. This could be attributed to the fact that the serial architectures here are also stereo vision heads, which are heavier, due to the fact that they carry two sets of cameras. However, the vergence acceleration of the individual stereo cameras is also lower than the pan acceleration of the parallel architecture vision platforms. The tilt acceleration of the mechanical MAVE is slightly higher than its pan acceleration, which is due to mechanical asymmetries of the camera mounting platform. However, the tilt acceleration is also within the mid range of the other five platforms. Overall, the table shows that the acceleration in all directions of the parallel architectures is superior to the acceleration of all directions of the serial architectures.

The pan velocity of the mechanical MAVE, at $930^\circ/s$, is the second highest in the table, after the Agile Eye with $1000^\circ/s$. The tilt velocity of the mechanical MAVE, at $913^\circ/s$, also follows the Agile Eye with $1000^\circ/s$. This slight difference in pan speed is again due to the fact that the camera mounting platform of the mechanical MAVE is asymmetric in the horizontal and vertical directions. The overall velocity performance of the listed active vision systems is again clearly separated by the performance of the parallel and serial architectures, where the parallel architectures are faster than the serial architectures.

The angular range of the mechanical MAVE in both the pan and tilt direction are lower than the angular range of any other joint on any other platform in the table. This is the result of using a gimbal to mount the camera and the use of linear motion to control movements. The other platforms all use rotational actuators that apply their force to a platform or a chain of kinematic links. In Yorick, the use of rotational actuators has even allowed the cameras to be rotated by full 360° around each controllable axis.

The positioning resolution of the mechanical MAVE could not be determined from its mechanical properties, as the actuators do not have a defined mechanical positioning accuracy. In theory it may be possible to generate a position resolution that is arbitrarily small. However, it

would be near impossible to implement a control mechanism that was able to control an infinitely small positioning accuracy. Manual positioning tests did show that it was possible to achieve a repeatable positioning accuracy of $< 0.068^{o}$. This was considered to be a valid angular resolution if it was possible to repeat this positioning step at least five consecutive times. This test raised the related question of long term positioning repeatability. A number of influences could alter this resolution value in the future, such as the type of controller used, the amount of performance feedback provided to the controller, the ambient temperature, fatigue and many other factors. The influence of ambient temperature on the mechanical MAVE has already been tested and was covered in section 3.6 and figure 3.16. Repeatability information on the other hardware systems was not readily available from the literature and could hence not be included in the table.

This short performance comparison of the mechanical MAVE with other, similar platforms has shown that the mechanical MAVE is certainly capable of keeping up with the performance of other systems. It was even shown that the mechanical MAVE can outperform most platforms on certain performance characteristics, but also be outperformed on others, such as angular range. However, it was not the aim to build a system that could outperform all other active vision systems. It was the attempt to build a cheap platform with off the shelf components that is capable of reproducing behaviours which are present in the human extra-oculomotor system.

6.2 Laser Target Controllers

The camera controllers covered in this chapter fall into one of three categories[1]: saccadic, smooth pursuit or a combination of both. Each of these control behaviours assumes distinctly different target behaviours. The test environment is therefore configured to produce target patterns that can accommodate the requirements of the camera controllers.

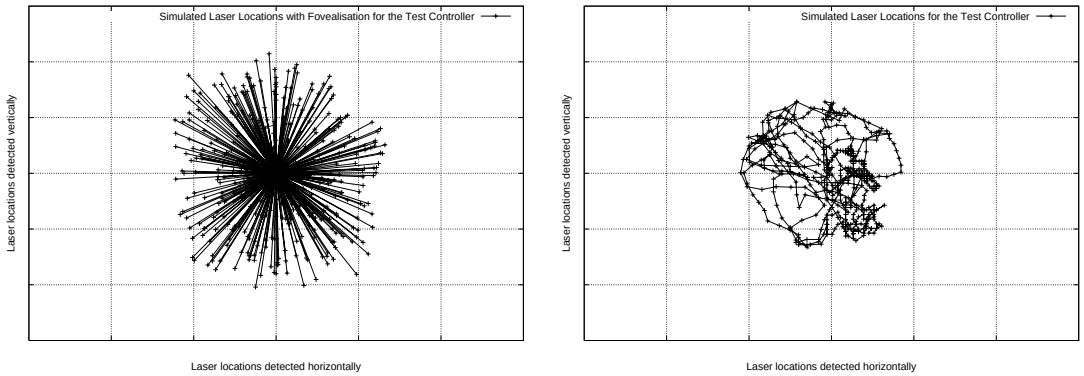


Figure 6.1: These images show 500 iterations of saccadic and smooth pursuit test patterns, generated by laser target control programs and recorded by the stationary MAVE, in simulation. The saccadic target pattern on the left produces locations that have a Gaussian distribution around the origin of the pattern. The smooth pursuit target pattern on the right changes the target velocity every 100 iterations.

[1] An exception being the stationary benchmark controller.

One target pattern generates saccadic targets with a Gaussian distribution. This is used for saccadic camera controllers. The other pattern produces a random target track with cyclic velocity fluctuations, which is used for smooth pursuit camera controllers. Throughout this dissertation all experiments, either in simulation or on the hardware, were conducted by using only these two target controllers. Figure 6.1 shows an example of these two target patterns, which represent only 500 target locations each, in contrast to the 50000 target locations that are produced during full trials. The full trial patterns are also initialised with the same random number seeds, in order to provide close to identical test conditions for each camera controller test. The following two subsections specify the algorithmic implementation of the two laser target controllers.

Before the target controllers are executed it is assumed that the laser mirrors have been configured to point the initial laser target into the fovea of the camera, when the camera is pointing straight ahead at the projection screen. This location forms point $0,0$ of an imaginary cartesian coordinate system, around which further target locations are generated.

6.2.1 Saccadic Laser Target Controller

The saccadic target controller uses five constants to control its operation. These constants specify the screen area on which patterns are generated and the proximity between locations that are still visible to the camera:

- *maxX* is set to 32 and specifies the maximum *x* position of target locations in the cartesian coordinate system.
- *minX* is set to -10 and specifies the minimum *x* position of target locations in the cartesian coordinate system. The distance from $(0,0)$ to $(minX,0)$ is smaller than from $(0,0)$ to $(maxX,0)$. This is due to the fact that the laser scanner is not set up perpendicular to to the projection screen, see figure 3.1.
- *maxY* is set to 25 and specifies the maximum *y* position of target locations in the cartesian coordinate system.
- *minY* is set to -25 and specifies the minimum *y* position of target locations in the cartesian coordinate system.
- *vr* is set to 50 and specifies the radius around the camera fovea in which target locations can be securely detected. This guarantees that a generated target location can be seen by the camera. *vr* assumes that the foveation on the previous target location was successful, as there is no feedback from the camera image to the laser target controller.

The algorithm operates as follows:

1. Define a random number generator rand$(0,1)$ that produces values in the range $[0.0,1.0]$.
2. Initialise the vectors **p** and **pLast**:

$$\mathbf{p} = \begin{pmatrix} maxX \\ maxY \end{pmatrix}, \qquad \mathbf{pLast} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

3. Calculate an orientation angle θ:

$$\theta = 359 \text{ rand}(0,1)$$

4. Calculate a length value l:

$$l = vr - \left(vr \exp\left(-\frac{8 \text{ rand}(0,1)}{27} \right) \right)$$

5. Calculate the target vector $\mathbf{p}'$:

$$\mathbf{p}'_x = \mathbf{pLast}_x - (int)(l \ \sin(\theta))$$
$$\mathbf{p}'_y = \mathbf{pLast}_y - (int)(l \ \cos(\theta))$$

6. Test the validity of $\mathbf{p}'$:

$$\begin{aligned} &\text{IF} \quad (maxX > \mathbf{p}_x > minX) \quad \text{AND} \\ &\qquad (maxY > \mathbf{p}_y > minY) \quad \text{AND} \\ &\qquad (l > \frac{vr}{100}) \\ &\text{CONTINUE} \\ &\text{ELSE} \quad \text{go to step 3} \end{aligned}$$

7. Move the laser mirror for horizontal target movement by $\mathbf{p}'_x - \mathbf{pLast}_x$ and the laser mirror for vertical target movement by $\mathbf{p}'_y - \mathbf{pLast}_y$.

8. Update the last position information $\mathbf{pLast} = \mathbf{p}'$.

9. Go to step 3.

6.2.2 Smooth Pursuit Laser Target Controller

Similar to the saccadic target controller, the smooth pursuit target controller also uses five constants to control its operation. These constants specify velocity parameters that define the speed of the target and a decay that keeps the target in a restricted area of the screen.

- *maxV* is set to 5 and specifies the maximum velocity of the target.
- *minV* is set to 1 and specifies the minimum velocity of the target.
- *velC* is set to 100 and specifies after how many iterations the target velocity should change.
- *error* is set to 0.5 and specifies a rounding error for the target speed. This helps take account of the fact that the laser target can only be positioned in fixed steps.
- *dec* is set to 0.96 and specifies a decay rate that encourages the target to return to the centre of the coordinate system.

The algorithm operates as follows:

1. As with the saccadic version, define a random number generator $\text{rand}(0,1)$ that produces values in the range $[0.0, 1.0]$.

2. Initialise the vectors and variables **p**, **pLast**, cV, i:

$$\mathbf{p} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \qquad \mathbf{pLast} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \qquad cV = minV, \qquad i = 0$$

3. Calculate a random vector **v** with values in the range [-1.0, 1.0]:

$$\mathbf{v}_x = 2.0\ \text{rand}(0,1) - 1.0$$
$$\mathbf{v}_y = 2.0\ \text{rand}(0,1) w_i - 1.0$$

4. Calculate a logarithmic multiplication factor f:

$$f = \sqrt{-2\frac{\ln(\mathbf{v}_x \mathbf{v}_y)}{\mathbf{v}_x \mathbf{v}_y}}$$

5. Calculate the target vector $\mathbf{p}'$:

$$\mathbf{p}'_x = (int)(2\ dec\ \mathbf{pLast}_x + \mathbf{v}_x f)$$
$$\mathbf{p}'_y = (int)(2\ dec\ \mathbf{pLast}_y + \mathbf{v}_y f)$$

6. Test the validity of $\mathbf{p}'$:

$$\text{IF} \quad (|cV| + error > |\mathbf{p}'| > |cV| - error)$$
$$\text{CONTINUE}$$
$$\text{ELSE} \quad \text{go to step 3}$$

7. Increment the counter i.

8. Test for velocity change:

$$\text{IF} \quad (i \mod(velC) = 0)$$
$$\text{THEN} \quad cV = cV + 1$$

9. Test for change in velocity direction:

$$\text{IF NOT} \quad (maxV > |cV| > minV)$$
$$\text{THEN} \quad cV = 1 - cV$$

10. Move the laser mirror for horizontal target movement by $\mathbf{p}'_x - \mathbf{pLast}_x$ and the laser mirror for vertical target movement by $\mathbf{p}'_y - \mathbf{pLast}_y$.

11. Update the last position information $\mathbf{pLast} = \mathbf{p}'$.

12. Go to step 3.

6.3 Stationary Benchmark Controller

After initial foveation, the stationary benchmark controller does not move the camera anymore. This allows all subsequent target locations to be measured with respect to the centre of the target test pattern. *Error Graphs* generated in this way provide information on the general distribution of targets throughout the target pattern. Although the saccadic and smooth pursuit patterns are fundamentally different, both sequences are designed to distribute targets over the same screen area. This provides a similar range of movement for the camera when interacting with either pattern.

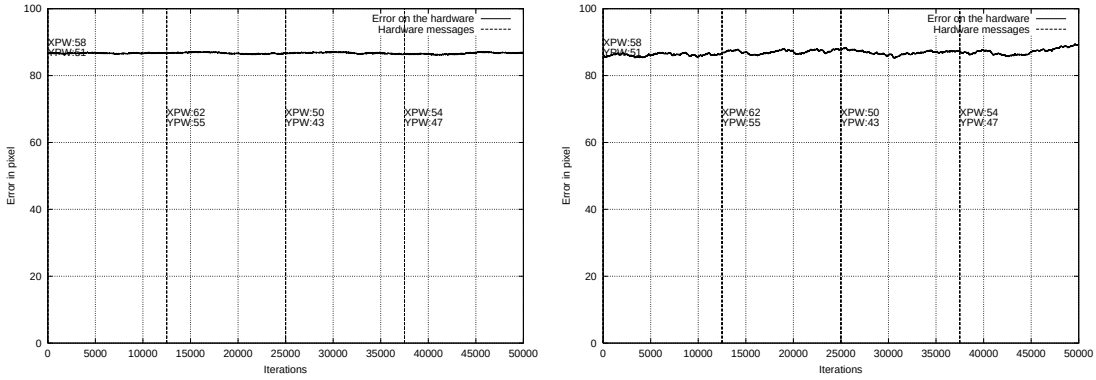


Figure 6.2: Error graphs for the stationary camera controller that keeps the MAVE pointing straight ahead and performs no camera activation to visual stimuli. For this test, the PW was changed every 12500 iterations (this has no impact on the camera movements here). Both images were generated on the robotic hardware by running the two laser target patterns. The left image shows the error graph generated by running the saccadic laser test pattern. The right image shows the error graph generated by running the smooth pursuit laser test pattern. Both graphs were smoothed by recursive averaging with a weight of 0.0001.

The similarly sized target area is evidently reflected in the error graphs of figure 6.2, where both curves show a similar error amplitude. The error graph of the saccadic target pattern in the left image of figure 6.2 and the error graph of the smooth pursuit target pattern in the right image of figure 6.2 have an average amplitude of 87 pixels. This value is fairly consistent over the 50000 iterations of each trial. Such steady error curves are fundamental to the objective assessment of the tested camera controllers. A gradually changing target behaviour would impact on the camera controller behaviour and very likely distort the results of the resulting error graphs.

6.4 Saccadic Controllers

The performance of both the adaptive and non-adaptive saccadic controllers described in sections 5.2 and 5.3 are covered in this section. Contrasting the behaviour of two different controller architectures that are designed to process the same type of target pattern in this section simplifies the comparison process. Figure 6.3 presents the error graphs of these two controllers. The left image is the error graph of the non-adaptive saccadic controller and the right image is the error graph of the adaptive saccadic controller.

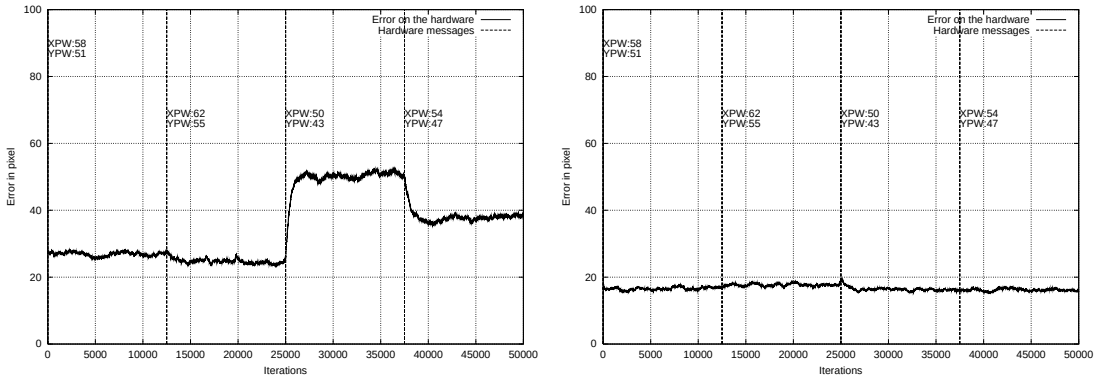


Figure 6.3: Error graphs for two variants of the saccadic camera controller. For this test, the PW was changed every 12500 iterations. The image on the left shows the error performance of the controller with a constant scaling factor. The image on the right shows the error performance of the controller with an adaptive scaling factor. Both graphs were smoothed by recursive averaging with a weight of 0.002.

It is apparent that PW alterations have a significant impact on the performance of the non-adaptive controller and virtually no influence on the performance of the adaptive controller. These observations confirm the design objectives of the controllers. The non-adaptive controller derives the camera activation time by multiplying a constant with the error measured between the fovea and the laser target. PW alterations have a significant impact on this control strategy. An increased PW produces longer saccades and a decreased PW produces shorter saccades. The impact of this behaviour is most apparent during the PW change from XPW:62 / YPW:55 to XPW:50 / YPW:43. Saccades that produce a relatively good foveation start to undershoot the target at this point and the error increases significantly.

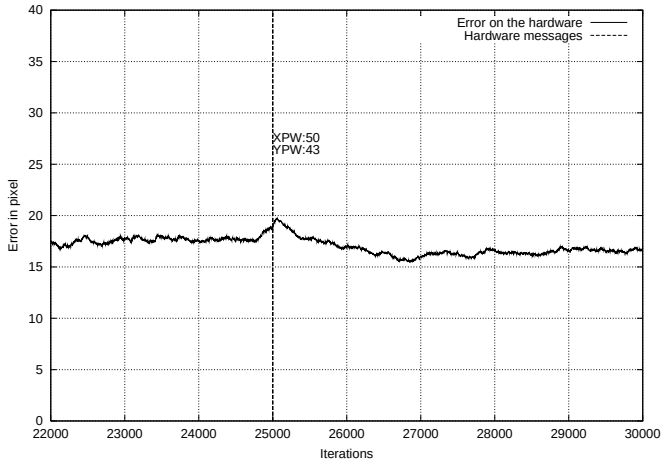


Figure 6.4: Expanded error graph for the adaptive saccadic controller from figure 6.3. The performance fluctuation is clearly visible during the PW change from XPW:62 / YPW:55 to XPW:50 / YPW:43. The graph was smoothed by recursive averaging with a weight of 0.002.

The adaptive saccadic controller appears to adapt well to the target pattern and PW changes, although it also shows a noticeable performance fluctuation at XPW:62 / YPW:55 to XPW:50 / YPW:43, figure 6.4. This very small fluctuation is due to the fact that a large and progressive weight adaptation is taking place to the multiplication factor that controls the activation time of the camera control. Compared to the non-adaptive saccadic controller, the adaptive controller can adjust the multiplication factor, depending on the quality of target foveation. This is a very simple and effective technique that can adapt very fast to behaviours of the camera dynamics and to small changes in the target behaviour. Larger and regular changes in the target behaviour would, however, cause a performance degradation, as the value of the multiplication factor would start lagging behind the measured performance of the target.

Although the controllers perform target foveation, and one of them is able to adapt its performance in response to PW changes, at no point does the error of either graph, in figures 6.3 and 6.4, settle to or approach 0. This is the result of two distinct influences which have already been discussed in previous chapters and are very specific to the robotic hardware and the camera controllers:

1. The amount of camera movement, with a constant activation time, is dependent on the camera starting position. This means that there is a non-linear relationship between the activation time and the distance the camera travels.
2. If there was a linear relationship between the activation time and the distance of the camera travels, there would still not be a factorial relationship between the activation time and the distance travelled.

Both of these influences are directly linked to the activation behaviour of the solenoids. Point 1 contributes inaccuracies into the positioning behaviour of the controllers, as there is no built in mechanism by which the rotational position of the camera can be measured and processed. Point 2 also introduces inaccuracies into the camera positioning behaviour as the activation distance is not proportional to the activation time, although this assumption is made by the controllers. Both of these points are covered in more detail in sections 6.6 and 7.3.

6.5 Smooth Pursuit Controllers

This section covers the adaptive and non-adaptive smooth pursuit controllers from sections 5.2 and 5.3, which operate in a similar way to the adaptive and non-adaptive saccadic controllers. The non-adaptive smooth pursuit controller multiplies a constant with the slip error between the fovea and the target to produce camera movement. The adaptive smooth pursuit controller also uses a multiplication factor to correct the slip error, but the multiplication factor is variable and is adjusted in accordance with the quality of the slip error correction.

Figure 6.5 shows the error graph of the non-adaptive smooth pursuit controller on the left and the error graph of the adaptive smooth pursuit controller on the right. Each curve demonstrates a very different behaviour, with the non-adaptive camera controller showing a very bad performance at XPW:62 / YPW:55. During this test phase, the controller falls into an oscillating behaviour and constantly overshoots the target. At one point, the error curve even overshoots the scale of the graph. This behaviour occurs because the increased PW, in combination with a fixed multiplication factor, produces camera activations that gradually overshoots the target during slip error correction.

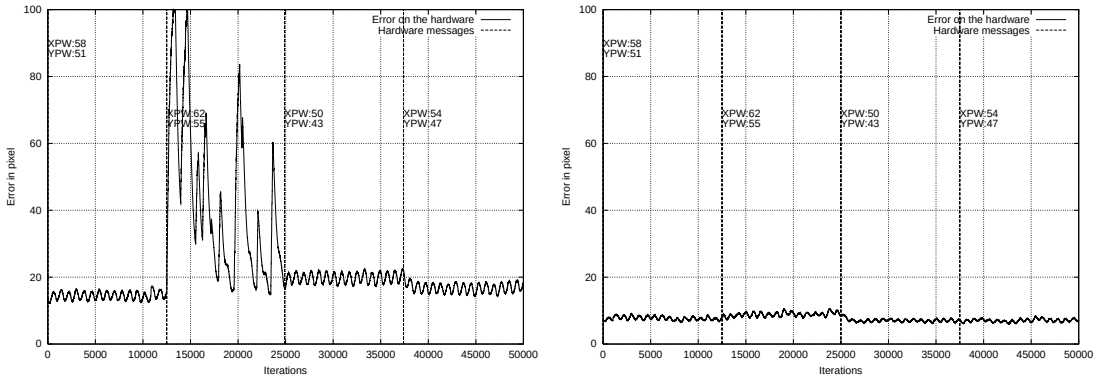


Figure 6.5: Error graphs for two variants of the smooth pursuit camera controller. The solenoid activation time was derived as the product of the error in pixels and a defined scaling factor. For this test, the PW was changed every 12500 iterations. The image on the left shows the error performance of the controller with a constant scaling factor. Oscillating target overshoots occur at a PW setting of XPW:62 / YPW:55. The image on the right shows the error performance of the controller with an adaptive scaling factor. Oscillations do not occur here. The graphs were smoothed by recursive averaging with a weight of 0.002.

As the target moves and the camera tries to correct the error again, the target is overshot again, this time producing an even greater error. This effect continues until the error increases sufficiently to cause the camera to oscillate from one side of the working envelope to the other. At the point during which the scale of the graph is overshot, target location data is also lost. This produces a gap in the stream of target location data, causing the iteration index numbers and the actual number of iterations in the graph to become misaligned. This phenomenon is most apparent in figure 6.6, where the PW changeover point and the 25000 iteration mark do not match up.

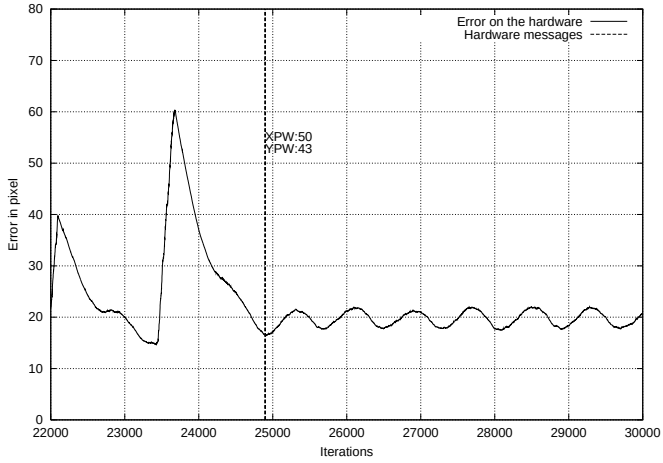


Figure 6.6: Expanded error graph for the non-adaptive smooth pursuit controller from figure 6.5. The last camera oscillation peaks are visible from the fluctuation period at XPW:62 / YPW:55. The regular error ripple after the PW change to XPW:50 / YPW:43 is also visible. The graph was smoothed by recursive averaging with a weight of 0.002.

The adaptive controller has an overall lower error curve and performs very well, even at XPW:62 / YPW:55. This suggests that the multiplication factor is adapting very fast to changes in the camera dynamics and the target pattern. The speed of factor change was also discussed in the last section and it was suggested that large and regular changes in the target behaviour would lead to a performance deterioration of this type of adaptive controller. The pursuit target pattern applies regular changes in form of cyclic velocity fluctuations. These are clearly visible as a near sinusoidal ripple in the error curves. The adaptive controller produces a lower amplitude ripple than the non-adaptive controller. This "smoothed" appearance, enlarged in figure 6.7, is due to the fact that the multiplication factor is rapidly adapting to the target velocity, while reducing the slip error during smooth pursuit. Even though this is an improvement over the non-adaptive controller, the adaptation still lags behind the actual behaviour of the target velocity changes.

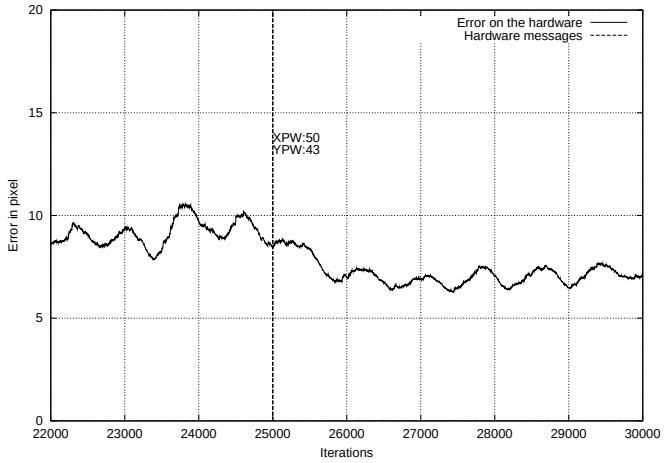


Figure 6.7: Expanded error graph for the adaptive smooth pursuit controller. The adaptation from XPW:62 / YPW:55 to XPW:50 / YPW:43 is visible. It becomes apparent how fast the adaptation takes place at this point. The graph was smoothed by recursive averaging with a weight of 0.002.

A more general observation shows that the error curves of both the smooth pursuit controllers are generally lower than those of the saccadic controllers. This is due to the fact that saccadic movements generally require longer camera movements for each sampling step than are required for smooth pursuit movements. This means that the relative error after saccades can be larger than after corrective smooth pursuit movements, the obvious exception being the oscillating phase of the non-adaptive smooth pursuit controller in figure 6.5.

6.6 Combined Least Squares Controller

The combined least squares controller integrates modified versions of the adaptive controllers that were tested in the previous two sections. This combination of control strategies and the ability to switch between them allows the controller to interact with a wider range of target behaviours. In order to test these controller capabilities, it is necessary to run a wider range of target behaviours than could be applied by either one of the pursuit or saccadic test patterns alone. The combined controller is therefore tested on both, the saccadic and smooth pursuit test sequence. This tests the performance of each control path and the ability to switch between them. While running each

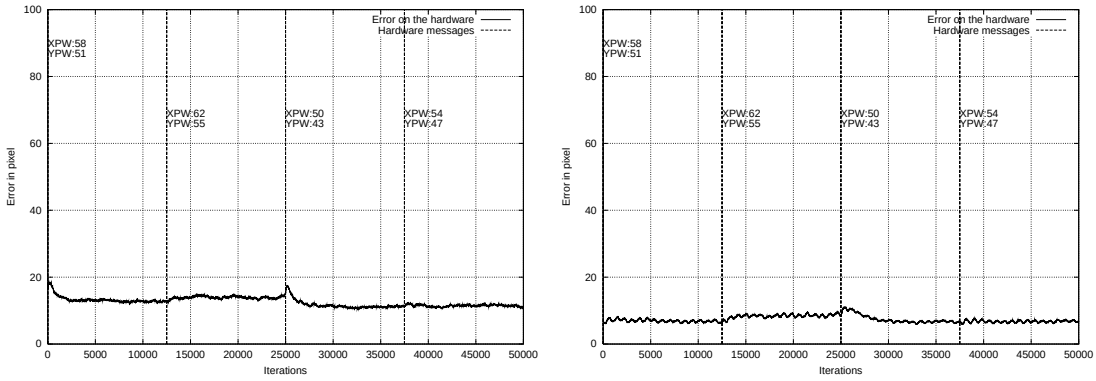


Figure 6.8: Error graphs that show the performance of the combined least squares controller in response to the saccadic and smooth pursuit test patterns. The solenoid activation times are derived as a product of error in pixels and polynomial results of activation functions. During each test, the PW was changed every 12500 iterations. The image on the left shows the performance under control of the saccadic test pattern. The image on the right shows the performance under control of the smooth pursuit test pattern. The graphs were smoothed by recursive averaging with a weight of 0.002.

of the two target sequences, the camera controller inevitably utilises both control paths, due to ambiguous target behaviours in both target patterns. However, the saccadic test pattern causes the camera controller to apply more saccadic control cycles than pursuit cycles; and the pursuit test pattern causes the camera controller to apply more pursuit control cycles than saccadic control cycles.

Figure 6.8 shows the camera controller performance under the saccadic test sequence on the left and under the pursuit test sequence on the right. Both error curves demonstrate a significant improvement to those of the non-adaptive saccadic and smooth pursuit controllers. The magnified error curve for the saccadic control path, figure 6.9, also shows an overall improvement to that of

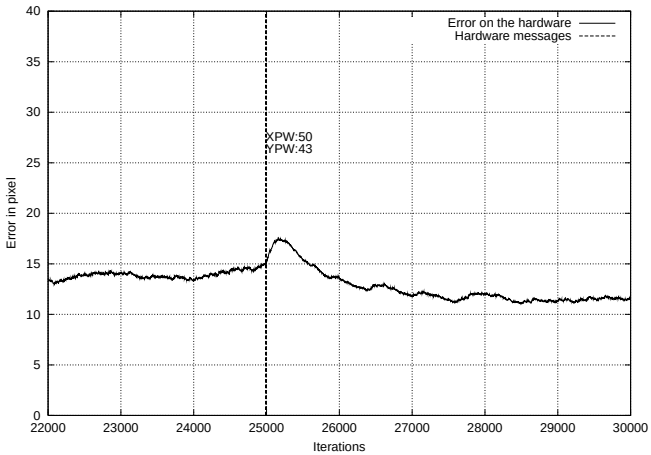


Figure 6.9: Expanded error graph for the adaptive saccadic path of the combined least squares controller. The prolonged adaptation period is shown after the PW change from XPW:62 / YPW:55 to XPW:50 / YPW:43. The graph was smoothed by recursive averaging with a weight of 0.002.

the adaptive saccadic controller, figure 6.4. This is not surprising, as the saccadic path of the dual controller allows up to one more saccade for each target location than is the case for the adaptive saccadic controller. Between XPW:62 / YPW:55 and XPW:50 / YPW:43 the error curve rises and indicates a slightly longer function adjustment period than is the case for the adaptive saccadic controller, see figure 6.9. This is due to the fact that the least squares approximation requires more data sets for an adaptation to take place than is the case for the single weight of the adaptive saccadic controller.

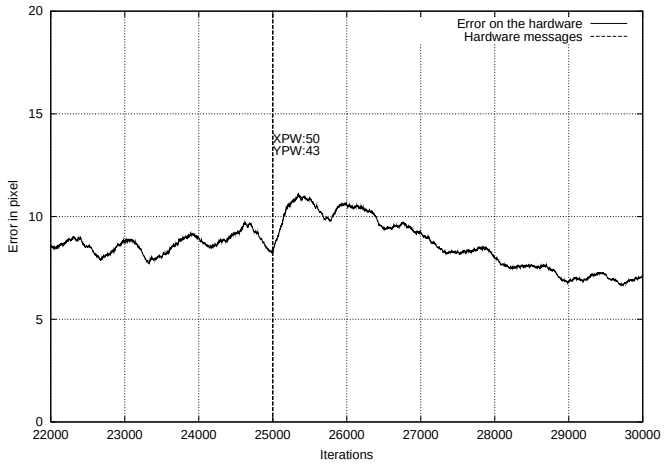


Figure 6.10: Expanded error graph for the adaptive smooth pursuit path of the combined least squares controller. The prolonged adaptation period is shown after the PW change from XPW:62 / YPW:55 to XPW:50 / YPW:43. The graph was smoothed by recursive averaging with a weight of 0.002.

The error curve for the pursuit path, figure 6.10 is very similar to the curve generated by the adaptive pursuit controller, figure 6.7, however there are small, but noticeable improvements. The overall error and the amplitude of the sinusoidal ripple are smoother. As the pursuit path of the dual path controller uses a function to represent activation weights, error measures that lie a number of iterations back can contribute directly to the calculation of activation weights. This means that the activation weight for a certain error at one point of the sinusoidal ripple can be calculated from values that were measured during a similar point of the prior sinusoidal ripple. As the adaptive pursuit controller only uses a factor to calculate the activation weight, only more recent measurements contribute to the calculations of the activation weight. The adaptation process of the combined least squares pursuit path is visible in figure 6.11, between the iterations 0-5000 and 37500-40000. At these sections, the ripple amplitude is initially high and gradually decreases. The least squares calculation adapts to the cyclic behaviour of the target pattern during these periods and starts to produce more accurate activation weights in response. It should be noted that such an adaptation behaviour is not specific to the pursuit control path and that the saccadic control path is equally capable of adapting to a fluctuating target behaviour.

These are very encouraging results, and as is stated in section 5.4, it can be assumed that this or a similar controller would indeed be a good candidate on which to base a gaze controller that can interact with the natural environment.

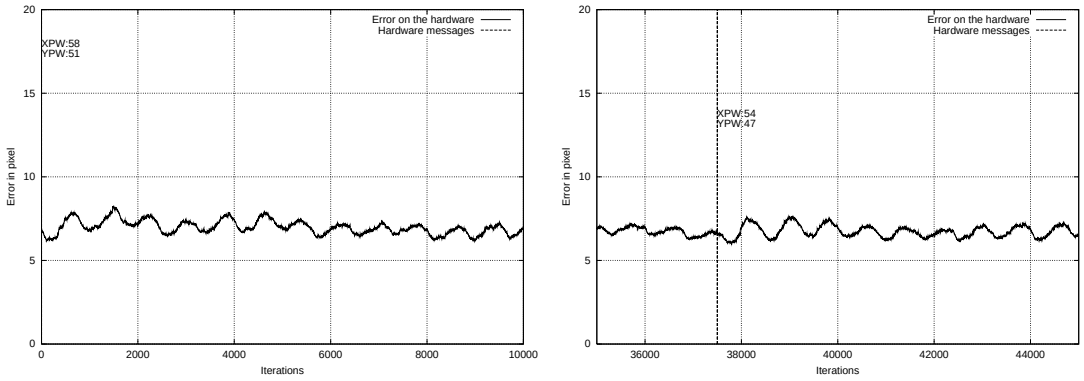


Figure 6.11: Expanded error graphs for the adaptive smooth pursuit path of the combined least squares controller. These graphs show the adaptation period after initialisation and PW change. The decreasing ripple amplitude indicates that an adaptation to the target velocity fluctuations is taking place. The graphs were smoothed by recursive averaging with a weight of 0.002.

6.7 Dog Net Controller

The *dog net* has a similar underlying control strategy to the controllers from the previously sections, but it uses a totally different implementation. This controller is based on the configuration of neurons in the brain and simulates processes that are believed to underlie saccadic control in the superior colliculus. The detailed properties of the controller are covered in section 5.5, but results and the transfer from the original platform to the MAVE have not yet been discussed. This section reproduces the results reported by Burdess [24] and discusses the modifications that were implemented to enable the controller to interact with the MAVE. The performance of the controller is then assessed, using the familiar error graph representation.

6.7.1 Reproduced Results

The original dog net was implemented in visual basic and ran on a 80486 DX2-66 processor. The program was equipped with a graphical interface that was able to visualise the state of the lattice and saccadic nodes. The positions of lattice nodes represented the locations of neurons in the upper layer of the superior colliculus, which form a topologically conserving map with the locations of excitation. Each node was connected to the neighbouring nodes by connecting lines. The positions of the saccadic weight nodes represented the locations of neurons in the lower layer of the superior colliculus. These nodes were mapped to the location of the lattice nodes. The activation weights of each saccadic node was represented by activation vectors that represent the distance and direction to the centre of the network.

In order to utilise this controller in the experimental environment, it was necessary to carry out two steps. First the original dog net had to be ported from visual basic[2] to a language which could be interpreted or compiled in the experimental environment of the MAVE. Second, it was necessary to modify the saccadic weight representation in such a manner that they could store activation

[2]The author was kind enough to provided a copy of the visual basic source.

times for MAVE. The natural choice of language was C/C++, as all other software components within the experimental environment are written in theses languages. Once the software port was completed, the correct operation of the controller was to be tested by comparing graphical network representations with the results reported by Burdess [24]. In the C/C++ implementation of the dog net controller, visual network configurations were produced by controlling a graph plotting program through a pipe. The results and the operation of the controller port are shown in figures 6.12, to 6.15. The node distributions and the vector weights have virtually identical configurations to those reported by Burdess [24], except for minor deviations, which are due to random factors in the target pattern. The integration of the controller into the experimental environment is covered in the following two subsections.

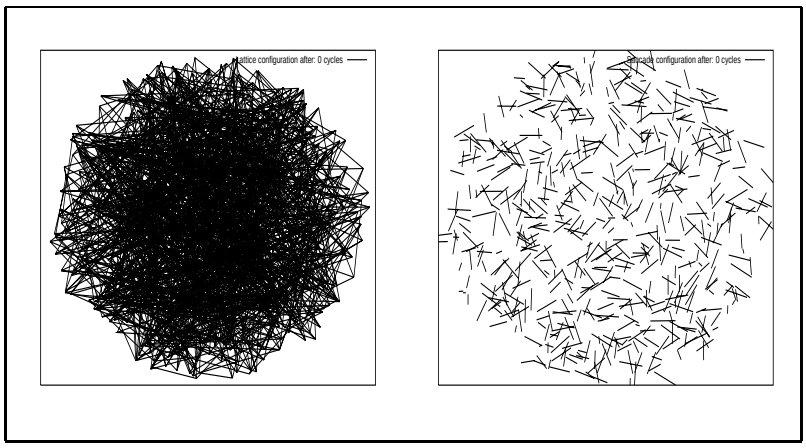


Figure 6.12: Lattice and saccadic weight distribution for the ported dog after initialisation and before any stimulus.

Figure 6.12 shows the network configuration before any stimulus has been presented. The left image shows the lattice configuration and the right image shows the saccadic weights associated with the individual lattice locations. The lattice nodes are randomly distributed over an area that represents the visual field. The length of the saccadic vectors is set to a random fraction of the radius of the visual field. This helps speed up the convergence of the network, but is not strictly necessary, as the convergence of the network would still occur for other values, provided there is enough time.

Figure 6.13 shows the network state after 4000 iterations. At this point the lattice nodes in the left image have started to form a web like configuration. The saccadic weights in the right image have also started to settle into a stable configuration that points the vectors from the lattice locations to the centre of the network. However, as the lattice nodes have not yet settled into fixed locations, it is important that the saccadic weights do not yet converge into a stable state. If the saccadic weights converged and pointed to the centre of the network before the lattice nodes settle into fixed locations, the saccadic weights would start to deviate from the centre, as the lattice locations change their positions.

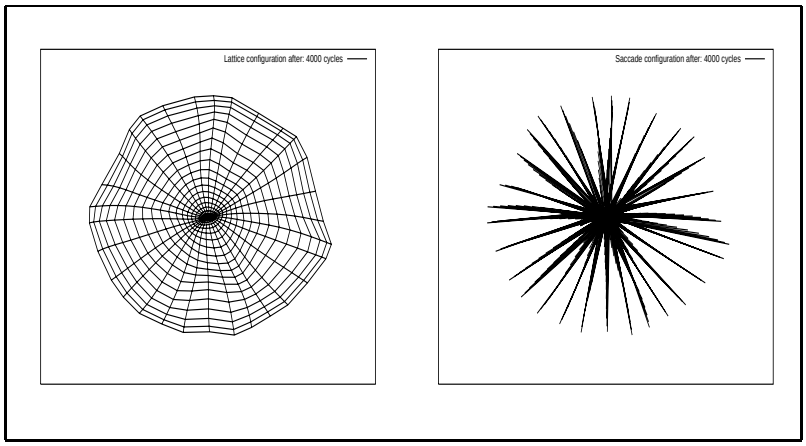


Figure 6.13: Lattice and saccadic weight distribution for the ported dog after 4000 iterations.

After 8000 iterations, figure 6.14, the lattice has expanded more and the network structure is becoming more stable. The saccadic vectors are now also more focussed on the centre of the network. From this state on, further iterations do not have much of an impact on the appearance of the network.

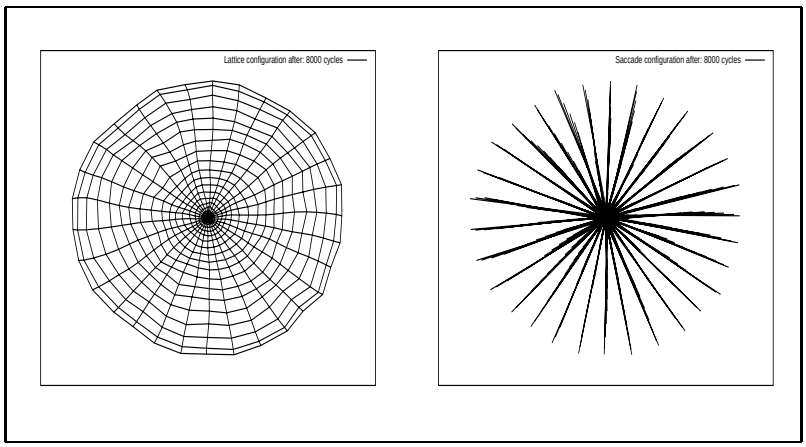


Figure 6.14: Lattice and saccadic weight distribution for the ported dog after 8000 iterations.

Figure 6.15 shows the network state after 16000 iterations. This network has not changed much since the network state after 8000 iterations (figure 6.14). At this point the vectors focus nearly perfectly into the centre of the network.

The four figures 6.12 to 6.15 replicate the results described by Burdess [24] and confirm the successful operation of the dog net in the C/C++ programming environment.

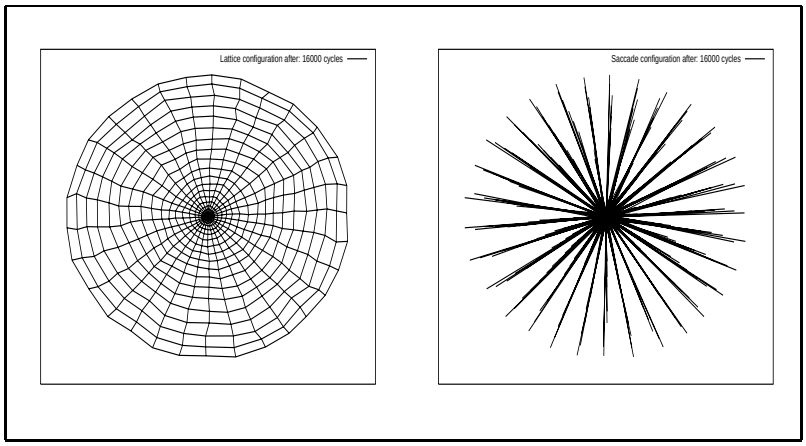


Figure 6.15: Lattice and saccadic weight distribution for the ported dog after 16000 iterations.

6.7.2 Hardware Performance

After the successful port of the dog net to the operating environment of the MAVE, the next step was to apply modifications that would allow the dog net controller to interface with the components of the experimental environment. This step was relatively straightforward, as the saccadic weight nodes could very easily be converted to represent activation vectors for the solenoids, and the cartesian target representation could very easily pass target locations to the lattice nodes. This is especially the case, as the dog net expects target locations to be represented in cartesian coordinates. Figures 6.16 to 6.19, show the lattice and activation weight configurations of the ported dog net, controlling the mechanical MAVE. Similar to tests of the original dog net, the test pattern has a Gaussian target distribution and the PW values remain fixed.

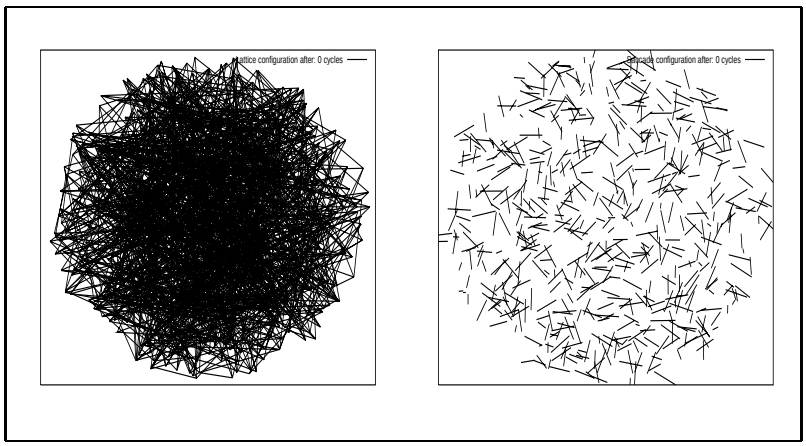


Figure 6.16: Lattice and activation weight distribution after initialisation for the ported dog net controller, configured to run on the robotic hardware.

Figure 6.16 shows the network before any training cycles have taken place. The weights are initialised in the same way as was done for the original dog net and the ported dog net in figure 6.12. Again, the magnitude of the randomised weight vectors is set to a small proportion of the radius of the visual field.

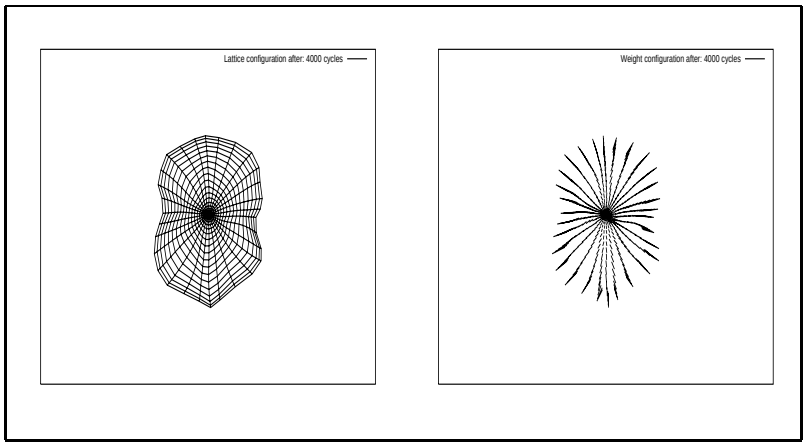


Figure 6.17: Lattice and activation weight distribution after 4000 iterations for the ported dog net controller, configured to run on the robotic hardware.

Figure 6.17 shows the state of the network after 4000 iterations. The lattice has started to form the familiar cobweb shape, although it is slightly distorted. This is due to the fact that the lattice locations are derived from cartesian target locations, which are generated by the laser target pattern. Due to physical restrictions of the gimbal on which the camera is mounted and target overshoots that lead the gimbal to hitting physical stop positions, the target pattern had to be compressed horizontally. This slightly modified target pattern does not appear to have any detrimental effect on the performance of the network, providing the area of the target pattern is not increased after the lattice nodes have converged to fixed locations.

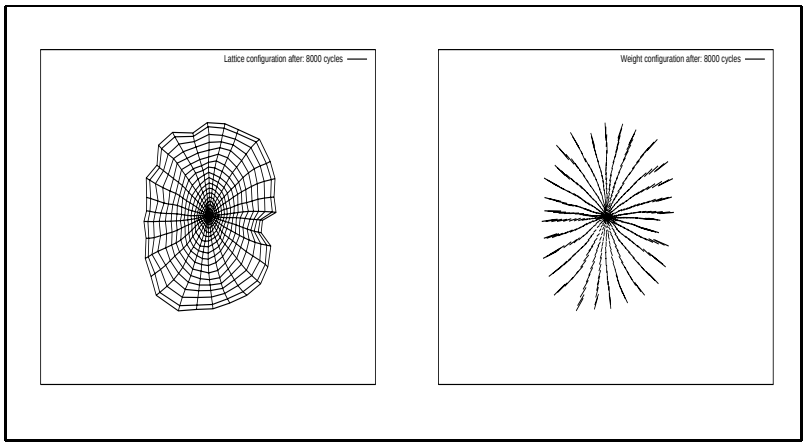


Figure 6.18: Lattice and activation weight distribution after 8000 iterations for the ported dog net controller, configured to run on the robotic hardware.

The solenoid activation vectors are also starting to form a pattern, pointing towards the centre of the network. They are noticeably shorter than the saccadic weights of the original dog net, and it could be argued that the weight vectors in figure 6.16 should be initialised to smaller values. However, it is evident that the network has no problem in adapting to this new configuration. In actual fact, this behaviour shows just how flexible the network is at adapting the activation weights.

Figure 6.18 shows the network after 8000 iterations. The lattice weights have expanded further, increasing the size of the network. At this point the network is also becoming more stable and progressive changes are happening on a much smaller scale. The activation weights are also aligning themselves more uniformly.

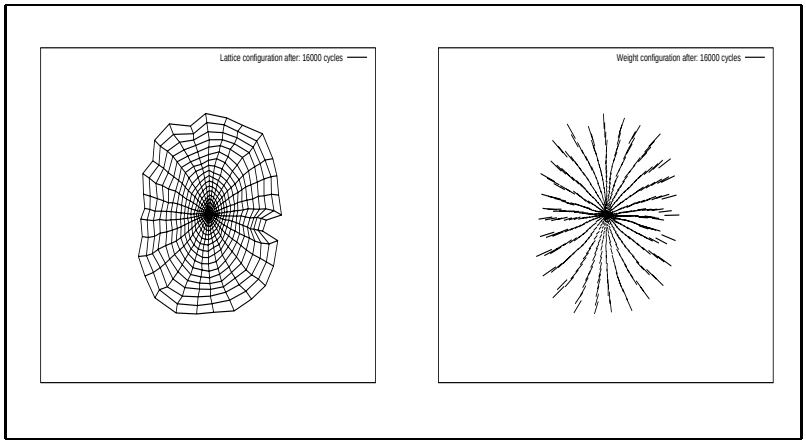


Figure 6.19: Lattice and activation weight distribution after 16000 iterations for the ported dog net controller, configured to run on the robotic hardware.

After 16000 iterations, the network in figure 6.19 has not changed much, compared to figure 6.18, and a stable state has been reached where further changes would be expected to be minor. This is the case, even if parameters in the experimental environment were to be altered, but just how well the controller has actually learned the dynamics of the MAVE cannot be interpreted from the lattice or saccadic weight figures. It is necessary to assess error graphs that are generated during and after the network has converged into a stable state.

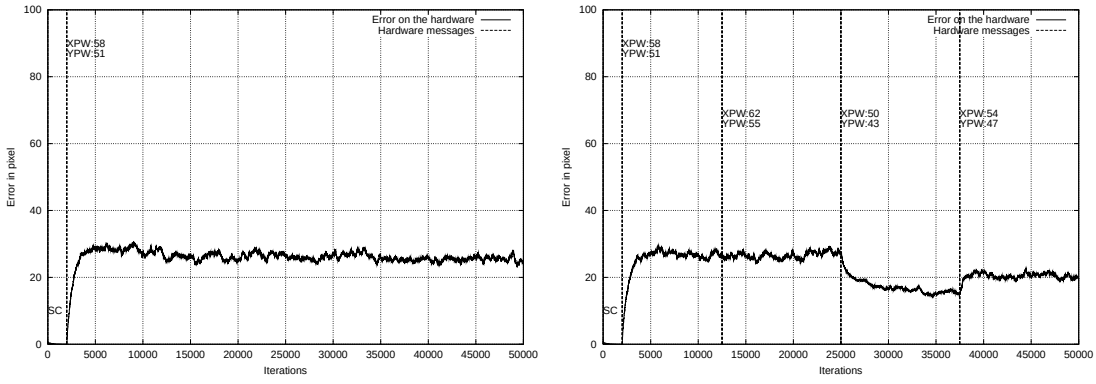


Figure 6.20: Error graphs for the ported dog net camera controller, running on the robotic hardware. The first 2000 iterations of the controller were run in simulation (**SC**), to prevent damage of the camera. The left image shows the controller performance at a continuous PW setting for all solenoids. The image on the right shows the controller performance as the PW changes every 12500 iterations. The graphs were smoothed by recursive averaging with a weight of 0.002

The error graphs in figure 6.20 show the controller performance without PW changes on the left and with PW changes on the right. In each graph the first 2000 iterations are marked with **SC**, which indicates that this part of the controller was trained in simulation. This serves purely to protect the hardware from damage, caused by initial random camera movements.

From the graph on the left, it is directly evident that the overall performance is not nearly as good as that produced by the adaptive saccadic controller in figure 6.3. This is surprising, as one could assume that the neural network would be very good at associating the error between the target and the fovea with an appropriate activation weight. Even if there was a factorial relationship between the error and the activation weight, the performance should not be worse than that of the adaptive saccadic controller.

The neural network has the potential to establish a better relationship between fovea to target errors and activation weights than could be achieved by the adaptive saccadic controller. However, the network learning rule makes one fatal assumption: It assumes a strictly linear relationship between the activation time and the amount of associated camera movement from **any** position within the working envelope. As this linear relationship does not exist within the experimental environment, correct activation weights cannot be learnt by the original dog net learning rule. To understand why this non-linearity produces such a performance degradation, it is first necessary to interpret the learning rule from equation 5.21. This rule performs the following operations: If two consecutive saccades, applied to one and the same target location produce a closer foveation than the first saccade alone, then replace the activation weights of the first saccade with the sum of the activation weights that produced the improved foveation. In a linear environment, this rule allows a very fast and precise weight adaptation, but in a non-linear environment, this learning rule generally learns to overshoot the target. The reason for this is illustrated in the following example:

A simplified, superficial situation is assumed in which the target to fovea error is identical for each new target location, and the camera starting position, from which the error minimisation is initiated is variable, requiring non-constant activation times for perfect foveations. In this environment, two possible situations are assumed:

1. The camera starts from a position from which a short activation time is required to achieve perfect foveation.
2. The camera starts from a position from which a long activation time is requires to achieve perfect foveation.

It should be noted that the network will try and adapt one and the same weight to achieve foveation in both cases. If it is assumed that the camera learns to perform perfect foveations with one saccade, under condition 1., subsequent applications of condition 2. would increase the activation weight, causing future conditions of 1. to overshoot. In this case, the second saccade may not contribute to an improvement and fail to invoke the learning rule, as a pair of saccades in opposite directions are less effective than two saccades in the same direction. This is partially due to the fact that the inertia of the moving camera in one direction needs to be reversed with return saccades. If it is now assumed that there is an equal distribution between the application of conditions 1. and 2., the activation weight will start to converge on a value that is more favourable to condition 2.

The error curve on the right hand side of figure 6.20 also confirms target overshoot learning. In this graph the PW is changed in an already familiar fashion, producing an initially surprising result. During XPW:50 / YPW:43 the error measure improves significantly, approximates and even exceeds the performance of the adaptive saccadic controller in the right image of figure 6.3. As was shown in figure 6.19, the network has very much converged after 16000 iterations and influences from the environment start to have a negligible impact on further network weight adaptations. This means that the adaptability of the neural network has come to a near standstill after 25000 iterations, at which point the PW is reduced and the performance improves dramatically. During PW settings of XPW:58 / YPW:51 and XPW:62 / YPW:55 the network learned to overshoot. The decreased PW of XPW:50 / YPW:43 requires larger activation times to cause foveation. It appears that the combination of a lower PW with previously excessive activation weights is now favorable to achieve good foveations.

Even though this network may be biologically inspired, it is hardly likely that it would operate in this way within the brain. As was discussed in subsection 2.2.2, there exists a low level feedback mechanism within the brain which provides a linearly controllable interface to oculomotor control. It could be the task of future research to implement such a mechanism on the MAVE. Section 7.3 covers such and other possible extensions in more detail. The problem of activation weight convergence appears to be more of a controller issue, as properties of the human eye do change over time and adaptability is essential. A possible solution to this problem is given in section 5.6 and tested in the following section.

6.8 Adaptive Dog Net Controller

The adaptive dog net controller from section 5.6 applies only a small modification to the decay mechanism in order to achieve non-convergence. The controller is tested under the same conditions as were present for the previous controller. The simulator is again used to run the initial 2000 training iterations.

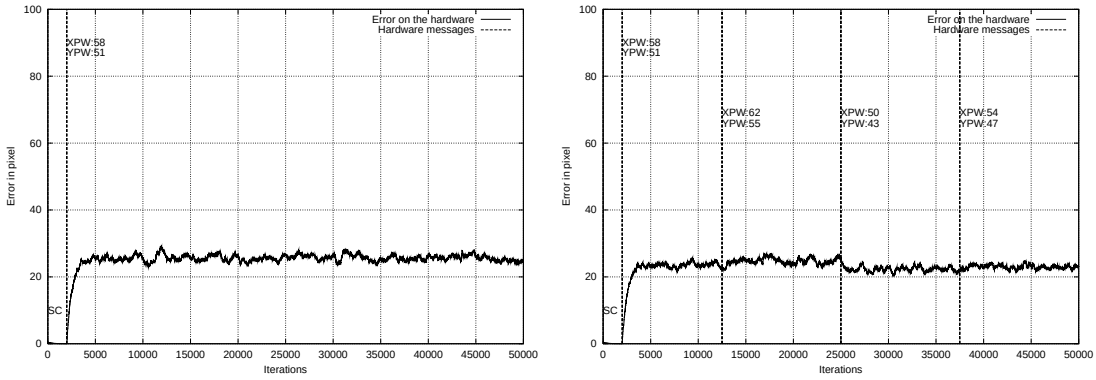


Figure 6.21: Error graph for the ported, adaptive, dog net camera controller, running on the robotic hardware. The first 2000 iterations of the controller were run in simulation (**SC**), to prevent damage of the camera. The left figure shows the controller performance at a continuous PW setting. The figure on the right shows the controller performance as the PW changes every 12500 iterations. The graphs were smoothed by recursive averaging with a weight of 0.002.

The left graph of figure 6.21 runs the controller without PW changes and the right graph runs the controller with PW changes. The general amplitude of the error graphs are nearly the same in both images. This also means that during XPW:50 / YPW:43, the error is worse than is the case in the right graph of figure 6.20.

6.9 Software Performance

		Error in Pixels			Error in Degrees		
		min	average	max	min	average	max
Saccadic Test Pattern Fig. 6.1L, 6.2L		**0**	**86.7075**	**167.8**	**0**	**8.31038**	**16.2616**
Saccadic Controller with constant scaling factor. Fig. 6.3L	XPW:50 YPW:43	0	50.5938	132.966	0	4.89726	12.5365
	XPW:54 YPW:47	0	37.601	111.041	0	3.63661	10.5365
	XPW:58 YPW:51	0	26.9336	120.354	0	2.58905	11.2841
	XPW:62 YPW:55	0	24.6513	137.993	0	2.37371	13.0732
Saccadic Controller with variable scaling factor. Fig. 6.3R	XPW:50 YPW:43	0	16.3145	107	0	1.57268	10.1266
	XPW:54 YPW:47	0	16.2329	109.142	0	1.5836	10.2461
	XPW:58 YPW:51	0	16.5984	119.08	0	1.61965	11.3576
	XPW:62 YPW:55	0	17.7734	111.018	0	1.73215	10.5247
Saccadic path of the Combined Least Squares Controller. Fig. 6.8L	XPW:50 YPW:43	0	11.5484	89.0505	0	1.19074	8.98902
	XPW:54 YPW:47	0	11.5012	96	0	1.20165	9.67723
	XPW:58 YPW:51	0	13.1422	167.443	0	1.33331	15.9445
	XPW:62 YPW:55	0	14.0369	86	0	1.42239	8.18076
Dog Net Cont. Fig. 6.20L	XPW:58 YPW:51	0	26.2676	223.081	0	2.61187	22.1897
Dog Net Controller Fig. 6.20R	XPW:50 YPW:43	0	17.0685	161.697	0	1.72197	15.7062
	XPW:54 YPW:47	0	20.3522	196.163	0	2.04417	19.4846
	XPW:58 YPW:51	0	26.698	216.187	0	2.6425	21.5034
	XPW:62 YPW:55	0	25.4454	216.058	0	2.52438	21.4774
Adp. Dog Net Cont. Fig. 6.21L	XPW:58 YPW:51	0	25.5855	301.496	0	2.55224	30.2122
Adaptive Dog Net Controller Fig. 6.21R	XPW:50 YPW:43	0	22.3856	199	0	2.25956	19.5258
	XPW:54 YPW:47	0	23.1006	200.022	0	2.32434	19.8815
	XPW:58 YPW:51	0	23.6603	203.062	0	2.35781	19.9271
	XPW:62 YPW:55	0	24.3852	208.062	0	2.4358	20.4433
Pursuit Test Pattern Fig. 6.1R, 6.2R		**0**	**87.4233**	**175.149**	**0**	**8.45204**	**17.3158**
Smooth Pursuit Controller with constant scaling factor. Fig. 6.5L	XPW:50 YPW:43	0	19.9132	61.0737	0	1.194752	5.64673
	XPW:54 YPW:47	0	16.6868	57.0351	0	1.64888	5.57873
	XPW:58 YPW:51	0	14.5384	144.779	0	1.43976	15.8474
	XPW:62 YPW:55	0	42.9689	285.237	0	4.22497	28.3549
Smooth Pursuit Controller with variable scaling factor. Fig. 6.5R	XPW:50 YPW:43	0	7.03635	80	0	0.771196	7.52731
	XPW:54 YPW:47	0	7.04804	61.0328	0	0.781032	5.6247
	XPW:58 YPW:51	0	7.72038	95.0474	0	0.845439	9.00191
	XPW:62 YPW:55	0	8.87054	80.5047	0	0.95034	7.60443
Pursuit path of the Combined Least Squares Controller. Fig. 6.8R	XPW:50 YPW:43	0	7.38499	57.0789	0	0.810604	5.24841
	XPW:54 YPW:47	0	6.76976	72.9931	0	0.773742	6.86854
	XPW:58 YPW:51	0	6.93287	65.2993	0	0.791212	6.36469
	XPW:62 YPW:55	0	8.4745	50.448	0	0.922744	4.64039

Table 6.2: A comparison of the performance characteristics for the controllers tested in this chapter. The values in this table were generated by the same data that was used to generate the error graphs in this chapter. References to the corresponding error graphs are provided in the table. It should be noted that the internal representation of the horizontal and vertical image resolutions differ. This causes slight deviations between pixel error and error in degrees.

Table 6.1 compares the hardware performance of the MAVE with the hardware performance of other active vision systems. Due to the mechanical nature of these other systems, software controllers can be designed to position the hardware to the accuracy specified in the table. As the MAVE does not have a mechanically defined positioning accuracy, it relies on the use of control algorithms to achieve a certain positioning accuracy. Table 6.2 compares the performance of the controllers tested in this chapter. The laser target controllers that produce the target information for the camera controllers are also included and are highlighted in bold. Their table values

were derived on the stationary benchmark controller from section 6.3. The error in pixels has been converted into error in degrees, to enable a comparison with other systems. As the internal representation of the horizontal and vertical image resolution are not the same, there are small fluctuations between the error in pixels and the error in degrees. This is not a calculation error, but due to the fact that each target position was individually converted from pixels to degrees.

As can be seen in table 6.1 and was discussed in the text following the table, the MAVE has a low positioning accuracy. Even though the positioning resolution of the MAVE, in table 6.1, is very low, it was derived under controlled conditions that were aimed at finding the best positioning resolution. The software controllers in table 6.2, on average, do not reach this positioning accuracy. In fact, the best saccadic controller achieves an average positioning accuracy that is 19.6 times lower than the MAVE positioning resolution from table 6.1. The average positioning accuracy of the best pursuit controller is only 11.34 times lower than the MAVE positioning resolution from table 6.1. In smooth pursuit mode, this means that positioning accuracy of the MAVE is 4284 times lower than that of Yorick, but only 5.7 times lower than that of the Agile Eye. When comparing these results, it should be kept in mind that the positioning resolution of the MAVE is not mechanically fixed. It may be possible to derive more effective software controllers in the future, that are capable of producing a much higher positioning resolution.

6.10 Summary

The hardware performance properties of the MAVE are contrasted with properties of other vision platforms at the beginning of this chapter and show that the MAVE exhibits encouraging behaviours. This is especially the case when considering that this vision platform is a prototype model, constructed mainly with off the shelf components.

The controllers tested thereafter form only a small range of conceivable control strategies, but the diversity of responses to the dynamics of the MAVE provide many interesting observations. The non-adaptive controllers are vulnerable to PW changes, so is the ported dog net, after convergence has taken place. It has also been shown that the learning rules can impact on controller performance. It has also emerged that the adaptive controllers perform most gracefully in a range of control conditions, with the combined least squares controller showing the best performance in both, the saccadic and pursuit environment.

Chapter 7

Conclusion

The objective of this investigation was to develop an active vision system that was very much inspired by the dynamics and performance of the human extra-oculomotor system. This was to allow the testing of low level control mechanisms that are not required in conventional active camera systems with position controlled actuators.

Chapter 2 started with a short review of vision and covered passive, active and purposive vision. This lead into a discussion of mechanisms that are involved in enabling and optimising active vision, such as gaze control, foveation, stereo and colour vision. Following this introduction, human and artificial active vision systems were discussed separately.

The anatomical investigation introduced four logical components of the human vision system: the eyeball, the protective apparatus, the motor apparatus and the visual pathway. The motor apparatus and the visual pathway were then covered in greater depth. This focused in particular on the behaviour and performance of eyeball positioning as a result of neural control and muscle contractions.

Artificial active vision systems were then introduced by first discussing fundamental mechanical design paradigms, this included: the common-elevation model, the independent gun-turret model, parallel architectures, serial architectures and types of actuation. Later individual design philosophies were discussed and the physical performance of five selected vision heads were contrasted. The control of artificial vision systems was covered by presenting control models that were implemented in the literature, including neural networks, finite state machines and control theoretical systems. It was suggested that new developments in artificial active vision could contribute to the understanding of biological vision and advances in artificial active vision alike.

Chapter 3 introduced the development of the Monocular Active Vision Eye (MAVE) and a laser targeting system. A design was devised for this purpose that incorporated physical properties of the human eye. This included static balancing, a parallel architecture and opposing linear actuation. The electrical control of actuators was also based on biological concepts, where activation potentials cause muscle contractions. After the completion of the requirements analysis and the basic design steps, mechanical and electrical components were selected that could fulfil the specifications of the model. Shock absorbers were used to simulate the damping of tissue around the eyeball. Linear actuators simulated extra-ocular muscles. A gimbal was used to simulate the

operational range of the eyeball in its socket and pulse width modulation (PWM) was used to represent electrical control signals sent to the actuators.

The targeting system was introduced as a position controlled laser pointer that generates intense light dots on a projection screen. The position of targets was controlled by surface reflective mirrors which were mounted on the drive shaft of step controlled rotational actuators. The performance of the electrical and mechanical assembly was then tested at the end of the chapter.

Chapter 4 discussed the four software levels that control the experimental environment and the simulator which simulates the mechanical MAVE and the laser targeting system. The implementation level allowed gaze, MAVE and laser target controllers to be developed. In this level it was also possible to set up timing files that control the operation of complete test sequences. The interface level provided a front end to either the simulator or hardware control. The control level contained the simulator and the operational end to the hardware device driver level. The device driver level was responsible for the control of the components within the robotic hardware. Comparison testing between the hardware and the simulator concluded the chapter.

Chapter 5 introduced the controllers that regulate the low level operation of the MAVE. A benchmark controller was discussed that can help assess the performance of the target patterns. The following controllers were non-adaptive and used an error measure between the target and camera position to trigger actuator activations. The controllers introduced thereafter formed extensions to the non-adaptive controller and became increasingly sophisticated; first by introducing adaptability and then by combining controllers for saccadic and pursuit control. The final controllers discussed were based on a neural network model of the superior colliculus and modification details were covered that allowed them to be integrated into the experimental environment.

Chapter 6 presented the performance results of the hardware and the controllers covered in chapter 5. First performance characteristics of the mechanical MAVE were compared to the five active vision platforms, discussed in chapter 2, then target pattern controllers were introduced and assessed by the benchmark controller. The performance of all other MAVE controllers was tested thereafter. A wide range of performance response behaviours was collected and illustrated in the form of error graphs. Graphical representations that highlighted the locations of lattice neurons and activation weights were also used to illustrate the operation of the saccadic neural controllers. The results of this chapter support the claim that simple controllers can be developed for vision systems like the MAVE that perform saccades, pursuit or a combination of both.

7.1 Contributions

The work presented in this dissertation makes the following contributions:

- An experimental environment was proposed and designed, which presents similar control requirements to those found in the human eye. In particular, the movement of the camera was to be controlled by pairs of opposing actuators that, similar to human muscles, applied their force in a linear direction. Just like muscles of the extra-oculomotor system, these actuators were also without a fixed positioning resolution and were to be vulnerable to performance fluctuations or fatigue. The control signals sent to the actuators were also to be of a similar nature to those of human activation potentials that control muscle contractions.

  This environment was to allow eye control algorithms to be developed and tested that normally form low level feedback loops or movement control within the human eyeball posi-

tioning system.

- A statically balanced mechanical Monocular Active Vision Eye (MAVE) was built with four linear actuators that apply their force in parallel to a camera mounting platform. These actuators had no defined positioning capability and were hydraulically damped. Their control was conducted by pulse width modulation (PWM), which was utilised to simulate controlled performance fluctuations and allowed controllers to regulate camera movements. A laser targeting system was also constructed that allowed the implementation of precise and repeatable target patterns that could be detected by the MAVE.

- Based on previous research, new control algorithms were designed that address two main control architectures: non-adaptive and adaptive, which were either capable of saccadic control, smooth pursuit control or a combination of both. These controllers were then tested in the experimental environment, where foveating or following camera movements could be produced in response to laser target patterns.

- An established neural controller, based on anatomical knowledge of the superior colliculus, was also tested in the experimental environment. Prior to the investigations undertaken here, this *dog net* had so far only been tested in simulation. The integration of the dog net into the experimental environment required only small refinements to the algorithm. Test results showed that this biologically plausible controller was capable of operating in a physical environment.

- From the evidence gathered in this dissertation, it was shown that artificial vision systems can be built and controlled without the need for exact actuator positioning capabilities. In fact, the use of cheaper and less accurate mechanisms have been shown to produce very high performance results and even outperform some of the highly engineered mechanical active vision systems, currently in use.

7.2 Limitations

The following limitations were present in this research:

- The number of controllers was necessarily limited. However, the progressive development from straightforward to more sophisticated systems gave insights into aspects that could be considered further in the development of future controllers for this type of application. For example, it was seen that the addition of adaptability was essential in order to deal with the physical dynamics of the MAVE, although experiments did show that the performance of non-adaptive controllers can produce limited results.

- The experimental environment was intentionally restricted, with a targeting and detection system that consisted of a highly controllable laser mechanism, rather than the ability to process real world images. This was essential for the implementation of predictable and repeatable test sequences that would allow an objective assessment of different camera controllers. However, for the development of applications that can process real world information, it is necessary to change the configuration of the experimental environment.

- The properties of the mechanical MAVE were also restricted to reduce the developmental cost and system complexity. This was most evident during the design phase of the mechanical MAVE, where it was decided to only include five physical properties that are also present in the human extra-oculomotor system, namely: a parallel architecture, non-position controlled actuation, opposing actuation, static balancing and pulse width modulation. The restrictions were not only implemented by limiting the number of anatomical properties to

be included in the design, but also by using mechanical systems that intentionally exhibit slightly different behaviours to those found in the extra-oculomotor system of humans:

- A gimbal was used to reduce the operation range of the eyeball from three to two degrees in the mechanical MAVE. This eliminated mechanical torsion which, under some conditions does occur during human eye movement. The control of torsion in the experimental environment would have required a significant increase in mechanical and electrical complexity, which would not have been feasible for this project.
- A particular benefit of two controllable degrees of freedom also lead to a reduction of required actuators. It was only necessary to use four actuators, for horizontal and vertical camera movements, compared to the six muscles of the human eye, which control pan, tilt and torsion.
- Fixed rods also extended from the actuators to the gimbal and formed a mechanical restriction and complexity reduction. In human vision, muscles and tendons are flexible and require extra control to prevent muscle slacking. The rods cannot slack and a control mechanism that may prevent slacking is not required and could not be tested.
- The mechanical MAVE also did not provide a mechanism by which gimbal position information could be fed back to MAVE controllers. In humans, eyeball position information is provided by muscle spindles.

7.3 Future Work

Active vision and the investigations in this dissertation span a wide range of disciplines. Hence further work could also take place in a combination of research areas. Suggested future work focuses initially on basic improvements that could optimise the performance of the existing experimental set up, such as improving the laser target positioning capabilities and reducing problems with the damping system. Thoughts are then directed at aspects that could play a central role in extending the existing capabilities of the experimental environment. In particular, advances are put forward that would allow mechanisms to interact with real world images and access gimbal position information. This starts with proposals for real world image processing algorithms, followed by intrusive modifications to the hardware that would enable camera position information to be extracted. This leads to work on controller issues that may enable the system to interact with the modified hardware. The final suggestion focuses on extensive modifications to the experimental environment, that would implement a range of biologically plausible features that could not be implemented in the current hardware version.

Targeting System

The laser targeting system implemented in this research uses two stepper motors to position surface reflective mirrors that control the direction of a laser beam. This is not a very fast mechanism, reaching speeds of ≈ 400 steps a second. As the execution of experiments is run on one computer and is fully automated, this does not pose much of a problem. However, it could be envisaged that the laser target control is operated by one computer and the operation of the MAVE is controlled by an other computer. In such a set up it may be interesting to test how fast algorithms and the hardware can respond to a certain target behaviour.

A small modification to the existing stepper motor controllers may be able to increase the target speed. The current system only switches the power supply to the motor windings on and

off. However, it may be possible to drive the motors faster by using short bursts of high current capacitor discharges. These discharges can create stronger magnetic fields to drive the motor. As the discharges are only of short duration, the motor windings would also not overheat and damage the motors.

A possibly even faster laser target control may be achieved by redesigning the laser targeting system and replacing the stepper motors with voice coils. Voice coils can be controlled by simply changing the voltage. They do not require multi-strand control like the stepper motors. It is also not necessary to find the starting position of the mirrors, each time the system is restarted.

A final and more radical change to the targeting system could see the projection screen being replaced by a video monitor or video projector. This would not only enable the control of a fast target dot, but could also be used to generate a range of visual environments that could be used to test image interpretation controllers. To ensure the uniform quality of the images captured, it may however be necessary to synchronise the camera capture rate with the refresh rate of the monitor or projector.

Damping

The mechanical damping used in the current version of the MAVE suffers from stiction and performance degradation as a result of ambient temperature fluctuation. In a modified version of the MAVE, it may be possible to remove the shock absorbers and apply damping directly to the moving parts of the gimbal, such as the joints and rod-end bearings. Silicone Grease has unusual viscous friction properties and can be applied directly to moving components. This type of damping may be sufficient to damp the movement of the gimbal and prevent the side effects that are present in the current system.

Image Processing

Although the existing targeting system of the experimental environment provides an excellent mechanism for the development and controlled testing of different controller architectures, it does restrict the implementation of controllers that utilise real world images to generate camera positioning. The first extension should therefore address this point and provide a mechanism by which controllers can be developed that interact with real world images. Such an extension would essentially allow the integration of existing image processing algorithms and gaze controllers that already exist and react to real world data. Such a change to the current experimental set up would be relatively straightforward, as it would literally require the removal of the projection screen and the implementation of a dummy laser target control algorithm. It would then be down to the discretion of the developer to select what real world information should be presented to the mechanical MAVE.

Gimbal Position Feedback

This extension should focus on implementing a gimbal position measurement mechanism. This would open entirely new control capabilities within the existing experimental environment, allowing additional control levels to be added that could process camera position information. This is very much in the spirit of this research as it is based on physiological knowledge of the human eye. It is well established that there exists a sensory feedback mechanism within humans that provides information about the degree of muscle contraction to a low level position control of the

eyeball. It is assumed that a self-regulating mechanism exists that provides a linear control interface to eyeball positioning. Such a linear relationship evidently does not exist within the current mechanical MAVE. However, early in the design of the mechanical MAVE, it was anticipated that camera position information could be beneficial to future research. For this reason, fixtures have been mounted to the gimbal that can accommodate potentiometers, figures C.17, C.18 and C.19. These produce a change in electrical resistance, depending on the constellation of the gimbal rings. Electrical connections to the interface board have also been reserved for this application, see pins 6-9 of the MAVE PORT in figure B.3. It will be necessary to ground the signals and add further resistors to prevent damaging the interface board. Software interface extensions would also have to be made to the files **eye.c**, **eye.h**, **eye_con.c.hme**, **eye_con.c.uni** and **eye_sim.c**, in order to process the information, resulting from the changing gimbal positions. Although the changes involve modifications at a range of levels, they are quite minor, but would contribute significantly to further research.

Linear Control

With the extension of the experimental environment to process gimbal position information, the next steps would see the implementation of control mechanisms that, similar to the human eye, can produce a linear relationship to the eyeball position. Such a control mechanism could be implemented in a range of ways. One could investigate a low level pulse width (PW) change adaptability that can provide an interface to higher controller levels, that are less adaptable, such as the *dog net*. It could also be investigated if controllers can be designed that have a more dynamic relationship to each other. It would indeed be very interesting to see how such controller combinations would interact and evolve to generate a MAVE positioning control that behaves very much more like that of the human extra-oculomotor control system.

Oblique Actuators

The final suggested extension and future work would involve extensive changes to the existing mechanical MAVE or the development of an entirely new system. The current experimental MAVE utilises the dynamics of a gimbal to prevent mechanical torsion, but in biological vision, torsion is controlled by a complex interaction of all six extra-ocular muscles. In order to test possible biologically plausible control mechanisms, it would be a rather challenging task to develop and build an artificial active vision system that used six linear actuators to position a camera. It may be expected that mechanical singularities will play a greater role in such a development than it has done here, and many issues not conceived during the design could be brought to light during the experimental development.

7.4 Final Words

The field of artificial active vision is very diverse and much of the work involved in developing vision systems has revolved around the design of highly engineered electrical and mechanical platforms, controlled by a wide range of gaze control algorithms. Often these do not consider control issues in a biologically plausible context. The research here has questioned that paradigm and has designed, built and controlled an active vision system that is based on physical properties of the human eye, with an inherently lower positioning accuracy than is present in most active vision

systems. It has also been shown that it is possible to control such a system to perform seeing tasks, such as saccades and smooth pursuit. The required control is not complex and the performance characteristics are very high, compared to other active vision platforms. It is hoped that this understanding will, in future, give rise to vision systems that are cheaper and mechanically less precise, but have exceptionally high performance characteristics. This is particularly of interest in an ever more competitive industrial world.

Appendix A

Software Interface Specification

This appendix covers all of the system procedures and functions that are accessible in the implementation level of the software hierarchy. The explanations form only a short guide, but should be sufficient to aid the implementation of programs with the examples provided in the source code.

Unless otherwise stated in the following text, integer functions will return the following values:

0 Function call encountered an error.

1 Function call completed successfully.

2 Function call unnecessary (this is possibly returned if a parameter is already set or nothing has changed since the test function was last called).

A.1 Scheduling

void SetSchedulingDelay (double *s***)** initializes the scheduler timing. The parameter *s* sets up the number of time slices to be available to the MAVE controller, before the laser scanner is given control. The laser scanner is not timed. Its control may take as long as is necessary to reach the next target location.

void SetSequencingLaser (int *times***)** sets up the number of *times* the laser controller is to be called, before calling the MAVE controller.

void SetSequencingEye (int *times***)** sets up the number of *times* the MAVE controlled is to be called, before calling the laser controller.

void SetSequencingEnvChange (int *times***)** sets up the number of scheduled iteration *times* after which a user defined procedure **MakeEnvChange (void)** is called. This procedure can be used to change the properties of the experimental environment without informing the laser or MAVE controllers. This option is designed to test controller adaptability under changing experimental environment conditions.

void RunTestSequence (void) tests the laser scanner sequence, without running the MAVE controller. This allows test patterns to be sampled and stored in a log file.

void RunScheduledSequence (void) starts the time dependent scheduler with the parameters defined in: **SetSchedulingDelay (double** *s***)**. The scheduler will terminate after either the laser or MAVE controller have set an exit flag.

void RunSequencedSequence (void) starts the procedure dependent scheduler with the parameters defined in: **SetSequencingLaser (int** *times***)** and **SetSequencingEye (int** *times***)**. The scheduler will terminate after either the laser or MAVE controller have set an exit flag.

void PauseScheduling (void) switches scheduling off, to perform commands that may normally be scheduled, when running in scheduled mode.

void ContinueScheduling (void) switches scheduling back on, after it was switched off with **PauseScheduling (void)**.

int TestController (void) is a mandatory function call and must be present in one of the top level program files. It contains the code that controls the camera movement during scheduled operations. If scheduled operations are not required, this procedure must still be defined, but can be added as a dummy. If this function returns 0, a flag is set to indicate that the test has been completed. Log generation is shut down correctly and the scheduler is instructed to terminate. This also means that the laser target pattern is terminated, possibly interrupting a cycle that would normally include auto repositioning of the laser mirrors. If the return value is $\neq 0$ and a call to **TestPattern (void)** also returns a $\neq 0$, **TestController (void)** is called again.

int TestPattern (void) is a mandatory function call and must be present in one of the top level program files. It contains the code that controls the laser scanner during scheduled operations. If scheduled operations are not required, this procedure must still be defined, but can be added as a dummy. If this function returns 0, a flag is set to indicate that the test pattern has completed its cycle. Log generation is shut down correctly and the scheduler is instructed to terminate. This also means that a possible auto repositioning of the laser mirrors can be completed. Return values $\neq 0$ indicate that the test pattern is still running.

void MakeEnvChange (void) is a mandatory procedure call and must be present in one of the top level program files. It contains the code that modifies the experimental environment. If environmental changes are not required, this procedure must still be defined, but can be added as a dummy.

A.2 Environment Configuration

int ControlMode (int *mode***)** switches between the simulator and hardware controller. It is possible to change the *mode* during operation but this should only be done with great care! The following modes are available:

0 Hardware control

1 Simulation control

A.3 Meteor Configuration

int OpenCapture (void) opens the frame grabber board for operation.

int SetMeteorParams (int *scale***)** sets the internal storage resolution for the captured images. The following *scale* values are available:

0 200×200

1 567×768

0 allows a larger number of images to be processed that have a low resolution. 1 reduces the image sampling speed, but increases the image resolution.

void CloseCapture (void) switches frame grabbing off and frees allocated memory.

A.4 Hardware Configuration

int SetPort (int *port***)** specifies which com *port* is connected to the control unit. The default *port* is 1.

int SetBusDelay (int *delay***)** sets up the *delay* time between communication signals with the control unit. The value depends on the speed of the computer. The default is 0.

int SetScannerDelay (int *delay***)** sets up the *delay* time between each step of the laser mirrors. This value depends on the speed of the computer. The default is 0.

int SetPulseWidthUnitSize (double *size***)** sets the activation time unit *size* for the solenoids. This value depends on the speed of the computer. The default is 1.

double CalculatePWUS (void) calculates a unit size value of 1 ms that can be passed to **SetPulseWidthUnitSize (double** *size***)**.

int InitConfiguration (void) must be called after the **Set*** commands and before any other commands! It initializes the control unit with the **Set*** parameters.

A.5 Laser Operations

int CalibrateShutter (void) places the laser shutter in the closed position. This routine should be called every time the hardware is first used, as it mechanically test the position of the shutter.

int Shutter (int *status***)** opens and closes the laser shutter. For $status \neq 0$, the shutter is open. For $status = 0$, the shutter is closed. The function relies on the shutter being calibrated first, as no measurements of the shutter position are made here.

int Laser (int *status***)** switches the laser on and off. For $status \neq 0$, the laser is switched on. For $status = 0$, the laser is switched off. It is recommended to leave the laser switched on and use **Shutter (int** *status***)** to hide and show the target.

A.6 Mirror Operations

int CalibrateMirrors (void) places the mirrors to point the laser straight ahead. The mirror calibration is accurate to within ± 5.5 steps ($\pm 0.825^\circ$) in the X direction and ± 3.5 steps ($\pm 0.525^\circ$) in the Y direction. A higher accuracy is not achievable, due to the degree of backlash within the reduction gears, see Section 3.2. A user interface is available which allows the laser mirrors to be fine tuned by hand.

void XMirror (int *steps***)** rotates one laser mirror in units of 0.15° in the X direction. *steps* determines by how many 0.15° units the mirror is rotated. Values > 0 rotate the mirror clockwise. Values < 0 rotate the mirror anti clockwise. The value 0 has no effect.

void YMirror (int *steps***)** rotates one laser mirror in units of 0.15° in the Y direction. *steps* determines by how many 0.15° units the mirror is rotated. Values > 0 rotate the mirror clockwise. Values < 0 rotate the mirror anti clockwise. The value 0 has no effect.

void Mirrors (int *xsteps***, int** *ysteps***)** allows both laser mirrors to be rotated simultaneously. Values > 0 rotate the mirrors clockwise. Values < 0 rotate the mirrors anti clockwise. The value 0 has no effect. This procedure implements mirror interpolation.

A.7 Monocular Active Vision Eye Operations

int XPulseWidth (int *width***)** sets up the pulse *width* for the actuators that control horizontal camera movements. Valid parameters are in the range -63 to 63. 0 sets the pulse *width* of both actuators to 0. Values between -63 and -1 set the pulse *width* for one actuator, and values between 1 and 63 set the pulse *width* for the other actuator respectively. 0 represents a pulse *width* of 10%. ± 63 represents a pulse *width* of 70%.

int YPulseWidth (int *width***)** sets up the pulse *width* for the actuators that control vertical camera movements. Valid parameters are in the range -63 to 63. 0 sets the pulse *width* of both actuators to 0. Values between -63 and -1 set the pulse *width* for one actuator, and values between 1 and 63 set the pulse *width* for the other actuator respectively. 0 represents a pulse *width* of 10%. ± 63 represents a pulse *width* of 70%.

void XActivate (int *time***)** activates the actuators that control the horizontal camera movement. Values > 0 control one actuator, values < 0 control the other actuator. The unit size is determined by **SetBusDelay (int** *delay***)**.

void YActivate (int *time***)** activates the actuators that control the vertical camera movement. Values > 0 control one actuator, values < 0 control the other actuator. The unit size is determined by **SetBusDelay (int** *delay***)**.

void Activate (int *time*1**, int** *time*2**)** calls **XActivate (int** *time***)** and **YActivate (int** *time***)** in parallel.

It should be noted that the simulator only supports the following pulse width ranges: 0, -63 to -43 and 43 to 63! The above procedures also prevent opposing actuators from being called simultaneously.

A.8 Image Capture Operations

int CaptureDMA (int *status***)** starts continuous direct memory access (DMA) frame grabbing. DMA is configured with **SetMeteorParams (int** *scale***)**. **CaptureDMA (int** *status***)** can be switched off with *status* $= 0$ and on with *status* $= 1$. **CaptureDMA (int** *status***)** must be switched on when **CaptureFrame (void)** is called!

void CaptureFrame (void) copies an image from the continuously updated memory to a temporary memory location. Please refer to **SaveFrame (char** $*Fname$**)** and **SaveToX (void)** to display images stored.

int CaptureFrameDot (int *threshold***, int** *dotMin***, int** *dotMax***)** copies an image from the continuously updated memory to a temporary memory location and detects the location of the laser target. *threshold* defines the cutoff value between the background grey level and the laser target grey level. *dotMin* defines the minimum size of a bright image region with an 8-connectivity that is allowed to represent a target. *dotMax* defines the maximum size of a bright image region with an 8-connectivity that is allowed to represent a target. Please also refer to **XIn (int** *cols***)**, **YIn (int** *rows***)**, **RingIn (int** *rings***)** and **WedgeIn (int** *wedges***)** for information on converting the target locations into integer values.

It should also be noted that this function calls **ProcessImage (imStruct** *imageData***)**. The implementer should not attempt to call **ProcessImage (imStruct** *imageData***)** from anywhere within his own code.

int SaveFrame (char $*Fname$**)** saves an image from the temporary memory location to a file. The size of the file is determined by **SetMeteorParams (int** *scale***)**.

int SaveToX (void) saves an image from the temporary memory location to an output stream on the terminal. The size of the display window is determined by **SetMeteorParams (int** *scale***)**. This operation does not yet work, due to a bug within the METEOR driver!

A.9 Image Processing Operations

The following structure definition is only relevant if image analysis and gaze control algorithms are to be implemented on the robotic hardware.

```
struct imStruct
{
  unsigned char imBuff;   // Pointer to the image buffer
  int           imCols;   // Number of image columns
  int           imRows;   // Number of image rows
  int           xPos;     // Detected X image position
  int           yPos;     // Detected Y image position
  bool          validPos; // Declare the derived position to be valid
};
```

imBuff, *imRows* and *imCols* are values set by the experimental environment and define the beginning and size of the image array. *xPos*, *yPos* are values that are to be set within the function **ProcessImage (imStruct** *imageData***)**. These values specify *X* and *Y* coordinates of the image, with dimensions *imRows* and *imCols*, which the MAVE controller uses to control camera positioning. *validPos* has to be set to *TRUE*, indicating that the values set in *xPos* and *yPos* are valid target locations. If **ProcessImage (imStruct** *imageData***)** returns *NULL*, any modification made to *imageData* is not considered by the system. If **ProcessImage (imStruct** *imageData***)** returns *imageData*, the functionality of the built in laser target detection function **CaptureFrameDot (int** *threshold***, int** *dotMin***, int** *dotMax***)** is overridden.

imStruct ProcessImage (imStruct *imageData***)** is a mandatory function call that must be present in one of the top level program files. This function is only accessed when *mode* in **ControlMode (int** *mode***)** is set to 0. The body of this function is defined in the *Implementation Level* and allows the user to implement image analysis and gaze control algorithms. If **ProcessImage (imStruct** *imageData***)** returns *NULL*, the following functions in this section will use target positions detected by **CaptureFrameDot (int** *threshold***, int** *dotMin***, int** *dotMax***)**. If *imageData* is returned, the following functions in this section will access the values *imageData.xPos*, *imageData.yPos* and *imageData.validPos*. If *imageData.yPos = TRUE*, the returned image coordinates form valid target locations. Otherwise, the system is informed that a valid target location was not detected. The values *imageData.imBuff*, *imageData.imRows* and *imageData.imCols* are provided by the experimental environment and provide information of the image storage location and its size, specified by rows and columns.

It should also be noted that this function is called by **CaptureFrameDot (int** *threshold***, int** *dotMin***, int** *dotMax***)**. The implementer should not attempt to call **ProcessImage (imStruct** *imageData***)** from anywhere within his own code.

int XIn (int *cols***)** returns the *X* location of the laser target, detected by **CaptureFrameDot (int** *threshold***, int** *dotMin***, int** *dotMax***)** or set by **ProcessImage (imStruct** *imageData***)**. The location of the laser target or set position is scaled to a cartesian image size with a horizontal resolution of *cols*. If the location of the laser target is not detected, the return value is 0.

int YIn (int *rows***)** returns the *Y* location of the laser target, detected by **CaptureFrameDot (int** *threshold***, int** *dotMin***, int** *dotMax***)** or set by **ProcessImage (imStruct** *imageData***)**. The location of the laser target or set position is scaled to a cartesian image size with a vertical resolution of *rows*. If the location of the laser target is not detected, the return value is 0.

int RingIn (int *rings***)** returns the *ring* location of the laser target, detected by **CaptureFrameDot (int** *threshold***, int** *dotMin***, int** *dotMax***)** or set by **ProcessImage (imStruct** *imageData***)**. The location of the laser target or set position is scaled to a log polar image size with a ring resolution of *rings*. If the location of the laser target is not detected, the return value is 0.

int WedgeIn (int *wedges***)** returns the *wedge* location of the laser target, detected by **CaptureFrameDot (int** *threshold***, int** *dotMin***, int** *dotMax***)** or set by **ProcessImage (imStruct** *imageData***)**. The location of the laser target or set position is scaled to a log polar image size with a wedge resolution of *wedges*. If the location of the laser target is not detected, the return value is 0.

A.10 Log Generation

int GenerateLog (int *setLog***)** generates a log file for $setLog \neq 0$. No log is generated for $setLog = 0$. The log is used to store information about the performance of camera controllers. The information can later be extracted in form of graphs or plots, using the parser commands.

void EndLocationToLog (void) tries to save the current laser target position to the log file. Other system statistics are automatically saved to the log file during system operation, if data logging is switched on.

int TimeStamp (void) tries to write a time stamp for an error graph.

int TimeStampMessage (char $*$*message***, int** *location***)** tries to write a time stamped *message* to the current graph *location*.

The name of the log file is defined at the top of *parser.c* and may be changed if required. If *parser.c* is modified, it is necessary to rebuild the environment.

A.11 Log Parsing

The following arguments are used when calling procedures in the parsing environment:

logFile is the name of the file that contains the data to be parsed. The default file name is: “eye.log”.

gnuFile is the name of a temporary data file, which is used for storing parsed data. The data is later piped to gnuplot.

psFile is the name of the postscript output file.

txtFile is the name of the text output file.

actuator determines the camera actuator for which the statistics should be compiled. The valid argument range is: 0 to 3.

plotFormat uses the standard gnuplot formats to set the style of plotted lines. Recommended arguments are: “dots”, “lines”, “linespoints” and “points”.

time specifies the time interval that is to be plotted.

rangeFrom specifies the start range for the error plot.

rangeTo specifies the end range for the error plot.

weight specifies the weight used during recursive averaging.

cS specifies the start of the time interval.

eS shifts the label plot range and scales it to the error plot range selected.

The procedures for parsing the log file:

void WriteLogFile (char $*logFile$**, char** $*fileString$**)** writes a string to the **logFile*.

void SimFovTrack (char $*logFile$**, char** $*gnuFile$**, char** $*psFile$**, char** $*plotFormat$**)** simulates exact camera foveation on the running laser test pattern.

void ConTrac (char $*logFile$**, char** $*gnuFile$**, char** $*psFile$**, char** $*plotFormat$**)** uses data which was generated by **RunTestSequence (int** *onTimes***)** in hardware control mode. **psFile* is a postscript file which shows the laser test pattern without moving the camera.

void SimTrac (char $*logFile$**, char** $*gnuFile$**, char** $*psFile$**, char** $*plotFormat$**)** uses data which was generated by **RunTestSequence (int** *onTimes***)** in simulator control mode. **psFile* is a postscript file which shows the laser test pattern without moving the camera.

void ConCamTrack (char $*logFile$**, char** $*gnuFile$**, char** $*psFile$**, char** $*plotFormat$**)** uses data which was generated by **RunScheduledSequence (void)** in hardware control mode. **psFile* is a postscript file which shows the laser test pattern with the camera under the control of a MAVE control program.

void SimCamTrack (char $*logFile$**, char** $*gnuFile$**, char** $*psFile$**, char** $*plotFormat$) uses data which was generated by **RunScheduledSequence (void)** in simulator control mode. **psFile* is a postscript file which shows the laser test pattern with the camera under the control of a MAVE control program.

void ConPWTime (char $*logFile$**, char** $*gnuFile$**, int** *actuator***, char** $*psFile$**)** uses data which was generated by **RunScheduledSequence (void)** in hardware control mode. **psFile* is a postscript file which shows a graph of pulse width against activation time in *ms*.

void SimPWTime (char $*logFile$**, char** $*gnuFile$**, int** *actuator***, char** $*psFile$**)** uses data which was generated by **RunScheduledSequence (void)** in simulator control mode. **psFile* is a postscript file which shows a graph of pulse width against activation time in *ms*.

void ConPWActi (char $*logFile$**, char** $*gnuFile$**, int** *actuator***, char** $*psFile$**)** uses data which was generated by **RunScheduledSsequence (void)** in hardware control mode. **psFile* is a postscript file which shows a graph of pulse width against activation times.

void SimPWActi (char $*logFile$**, char** $*gnuFile$**, int** *actuator***, char** $*psFile$**)** uses data which was generated by **RunScheduledSequence (void)** in simulator control mode. **psFile* is a postscript file which shows a graph of pulse width against activation times.

void Error (char $*logFile$**, char** $*gnuFile1$**, char** $*gnuFile2$**, char** $*gnuFile3$**, char** $*gnuFile4$**, char** $*psFile$**, char** $*plotFormat$**, bool** *time***, int** *rangeFrom***, int** *rangeTo***, float** *weight***, int** *cS***, int** *eS***)** is a powerful facility that evaluates the target to fovea error. This facility also uses the **MakeEnvChange (void)** command to evaluate the camera controller performance on a changing experimental environment.

void ConStats (char $*logFile$**, char** $*txtFile$**)** uses data which was generated by **RunScheduledSequence (void)** in hardware control mode. *txtFile* is a text file which contains a statistical evaluation of the camera controller performance.

void SimStats (char $*logFile$**, char** $*txtFile$**)** uses data which was generated by **RunScheduledSequence (void)** in simulator control mode. *txtFile* is a text file which contains a statistical evaluation of the camera controller performance.

A.12 Additional Operations

int Bell (int *volume***)** sets the volume of the internal buzzer. Legitimate values for *volume* are 0 to 255. 0 turns the buzzer off.

int Fovealise (int *iterations***, float** $factor$**, float** *error***)** a built in command that implements a simple saccadic camera controller. This causes the camera to foveate on the laser target. *iterations* determines how many times a saccade is to be applied. *factor* specifies the weight of each saccade. Values should normally be in the range: $0 > factor < 1$. *error* specifies the maximum error between the distance of the laser target and the centre of the camera image, before prematurely terminating the number of *iterations*.

int Trigger (bool *switchState***)** switches the built in trigger functionality. *switchState = TRUE* closes the trigger circuit. *switchState = FALSE* opens the trigger circuit. This enables external devices to be triggered during system operation. Triggering is performed by an Opto-coupler with the following specifications: open collector 50 *mA* at max 30 *VDC*. The minimum hardware conversion time is $800\mu s$. The time for software conversion has not been measured and may vary from system to system. Please also refer to appendix B for information on polarity. It should be noted that the trigger implementation overloads the functionality of some suggested hardware expansions.

Appendix B

High Level Circuit Diagrams

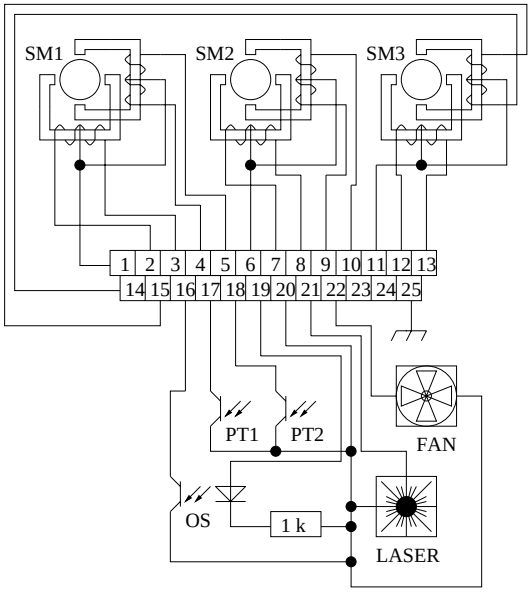


Figure B.1: High level circuit diagram of the laser scanner.

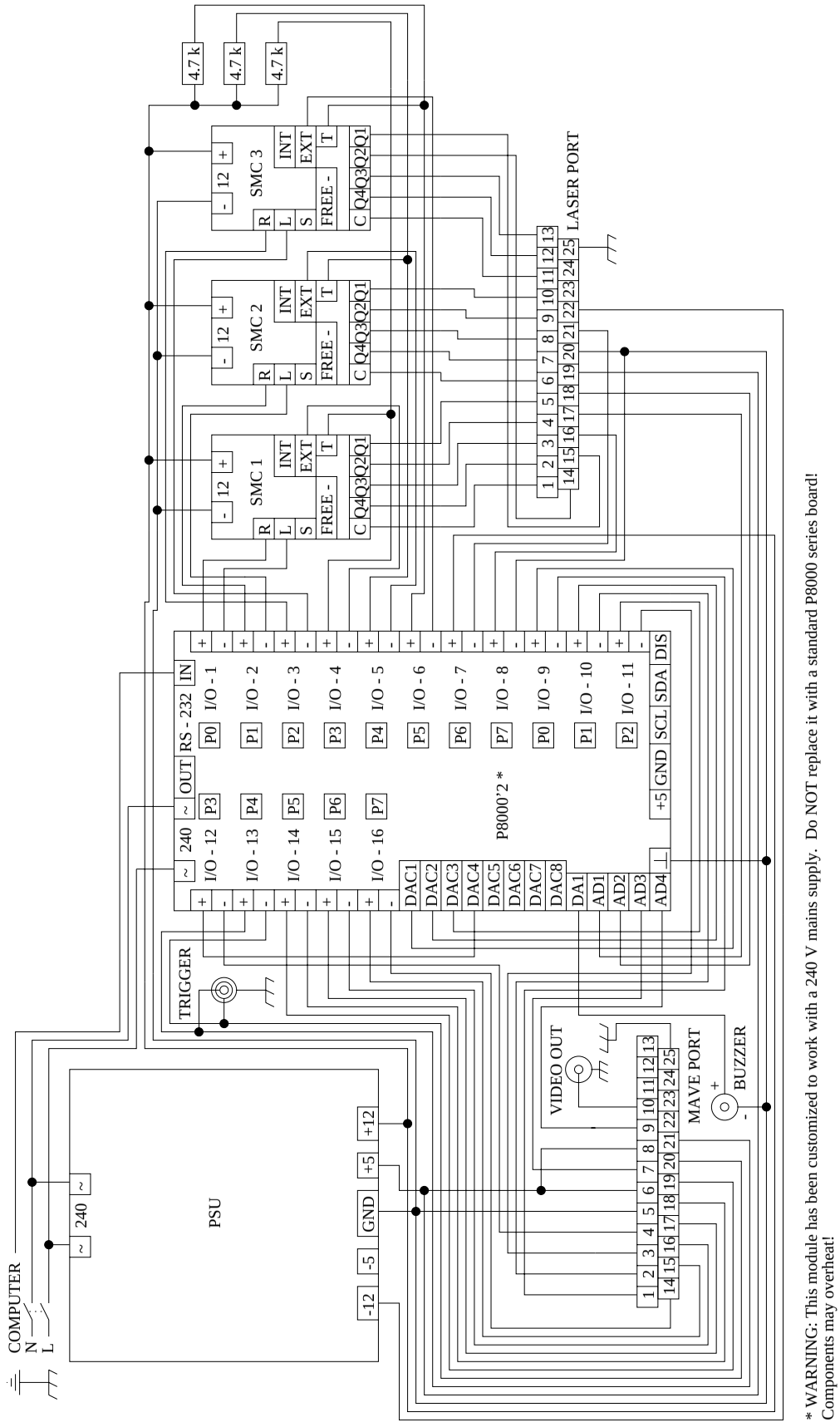


Figure B.2: High level circuit diagram of the control unit.

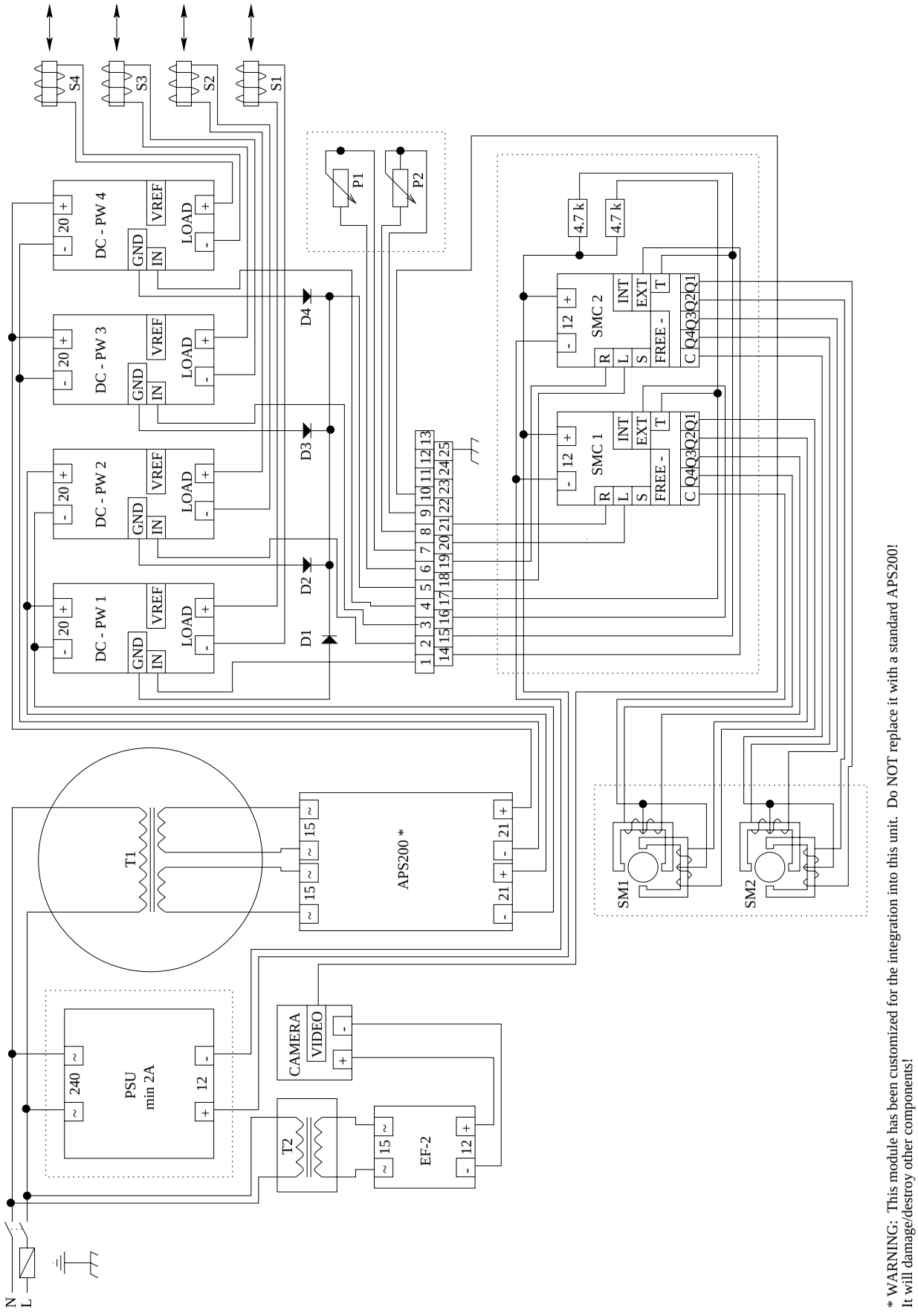


Figure B.3: High level circuit diagram of the MAVE.

Appendix C

Gimbal Assembly

The mechanical Monocular Active Vision Eye (MAVE) was primarily built from standard components, reducing the cost of production. However, an adequate camera mounting mechanism was not available and had to be manufactured specifically for the application. This appendix shows the manufactured components and provides a brief assembly plan that illustrates how the components fit together. The assembled camera mounting mechanism is referred to as the *gimbal assembly*, which itself is subdivided into the *gimbal ring assembly*, the *gimbal cross assembly* and the *gimbal frame assembly*.

C.1 Gimbal Ring Assembly

The gimbal ring assembly consists of two gimbal rings, four pins and four low friction polymer bearings.

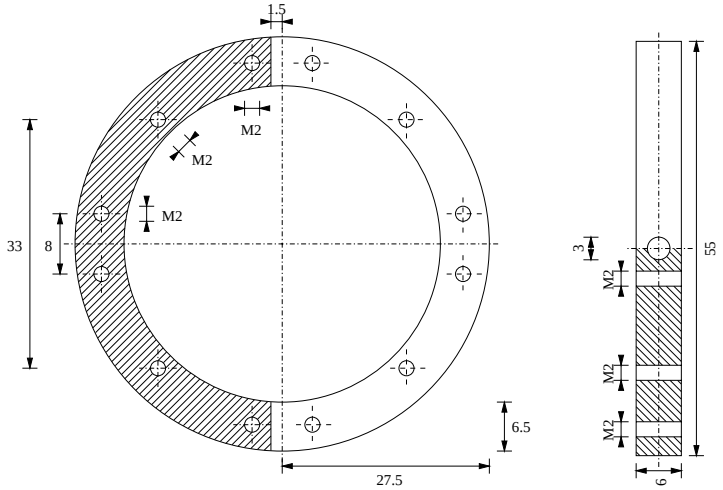


Figure C.1: Inner gimbal ring with taps to hold the gimbal cross and mount the CCD camera.

The inner gimbal ring, figure C.1, has twelve $M2$ taps and two $3mm$ holes. The taps provide attachment facilities for the camera and the gimbal cross assembly, figure C.10, to which rod-end

bearings are connected. The 3*mm* holes are used to fix 3*mm* pins which connect the inner gimbal ring to the outer gimbal ring.

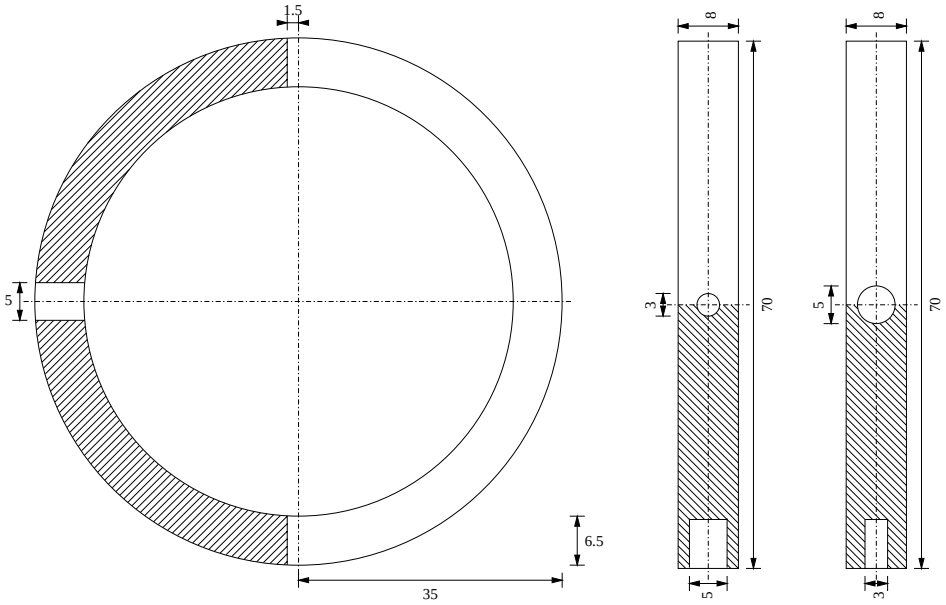


Figure C.2: Outer gimbal ring with holes for connection pins.

The outer gimbal ring, figure C.2, has two 3*mm* holes and two 5*mm* holes. The two 3*mm* holes are used to fix 3*mm* pins which connect the outer gimbal ring to the gimbal frame assembly.

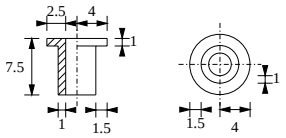


Figure C.3: Gimbal ring low friction polymer bearing.

The 5*mm* holes are fitted with low friction polymer bearings, figure C.3, and connect the inner gimbal ring with the outer gimbal ring.

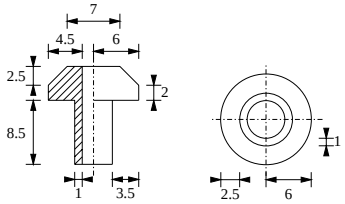


Figure C.4: Top plate low friction polymer bearing.

The top plate low friction polymer bearing, figure C.4, fits into the gimbal top plate, figure C.15, and connects through a 3*mm* pin to the outer gimbal ring.

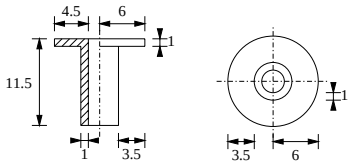


Figure C.5: Base plate low friction polymer bearing.

The base plate low friction polymer bearing, figure C.5, fits into the pedestal of the gimbal base plate, figure C.15, and connects through a 3*mm* pin to the outer gimbal ring.

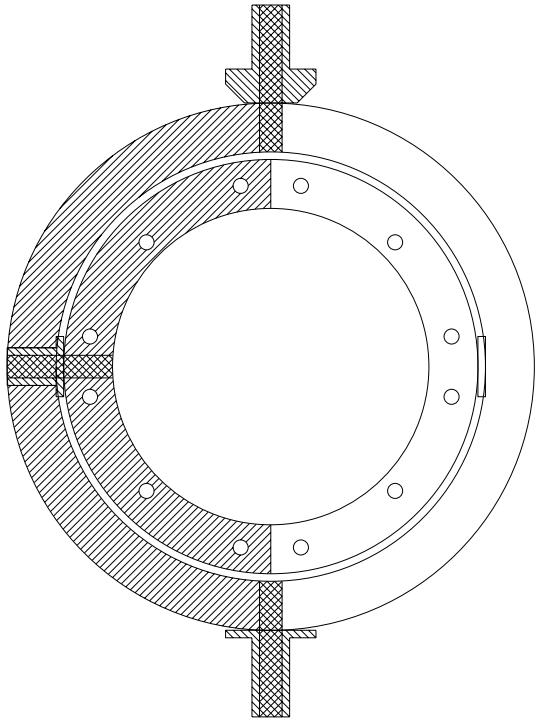


Figure C.6: Gimbal ring assembly with bearings and pins.

The gimbal ring assembly, figure C.6, shows the assembled gimbal rings with all bearings and pins. The outer gimbal ring can rotate freely in the base plate and top plate bearings and provides a platform with horizontal movement. The inner gimbal ring can rotate freely in the gimbal ring bearings and provides a platform with vertical movement. As the inner gimbal ring is mounted inside the outer gimbal ring, a fixed camera would be able to perform horizontal and vertical movements simultaneously.

C.2 Gimbal Cross Assembly

The gimbal cross assembly consists of two cross bars and four gimbal ring fixtures.

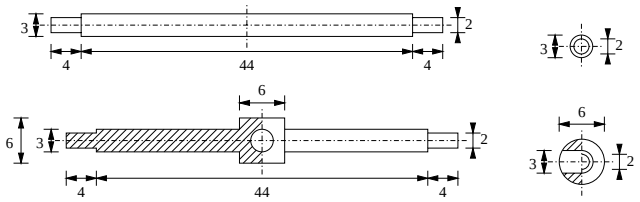


Figure C.7: Gimbal cross components.

The two crossbars, figure C.7, are solid cylinders with machined ends. One of the cylinders is widened in the middle to facilitate an assembly hole.

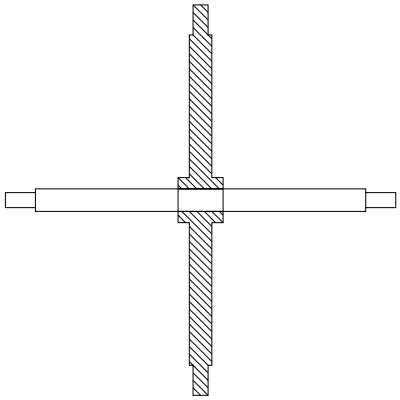


Figure C.8: Assembled gimbal cross.

The hole allows the crossbars to be assembled in the form of a cross, figure C.8.

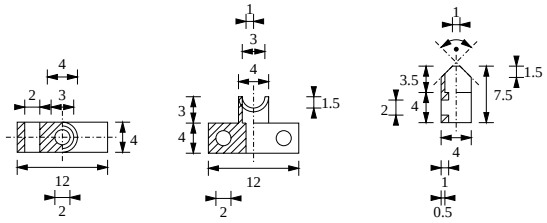


Figure C.9: Gimbal cross fixtures that allow the gimbal cross to be mounted to the inner gimbal ring.

Gimbal ring fixtures, figure C.9, have two 2*mm* holes and machined openings that hold the ends of the gimbal cross in place.

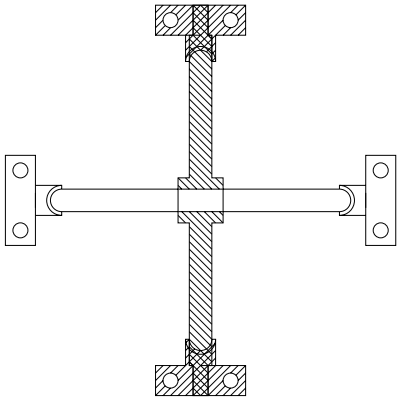


Figure C.10: Gimbal cross assembly with fixtures.

The gimbal cross assembly, figure C.10, has eight 2*mm* holes that allow the cross to be mounted on the inner gimbal ring.

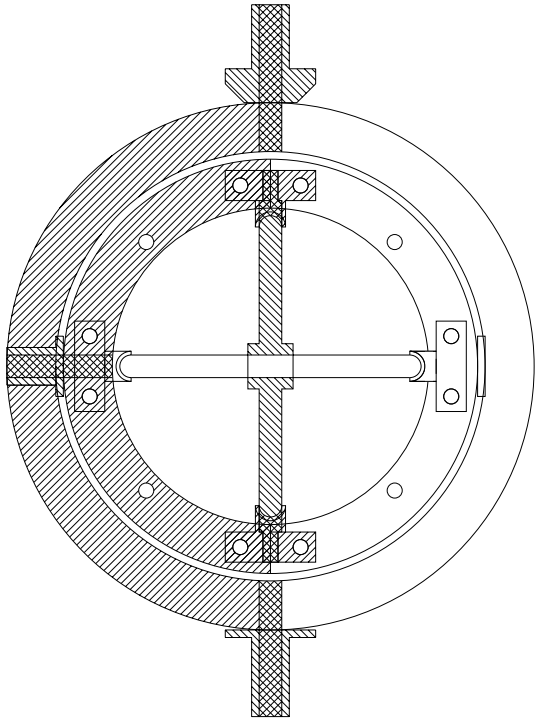


Figure C.11: Gimbal rings and cross assembly.

The Gimbal ring and cross assembly, figure C.11, shows the gimbal cross mounted on the inner gimbal ring. The eight 2*mm* holes of the gimbal cross align perfectly with eight *M*2 taps of the inner gimbal ring. It should be noted that the rod-end bearings are not shown in this image and are to be fixed to the gimbal cross before it is mounted on the inner gimbal ring. Spacers, not shown here, are screwed to the gimbal cross to hold the rod-end bearings in place.

C.3 Gimbal Frame Assembly

The gimbal frame assembly consists of the gimbal base plate, the gimbal back plate and the gimbal top plate.

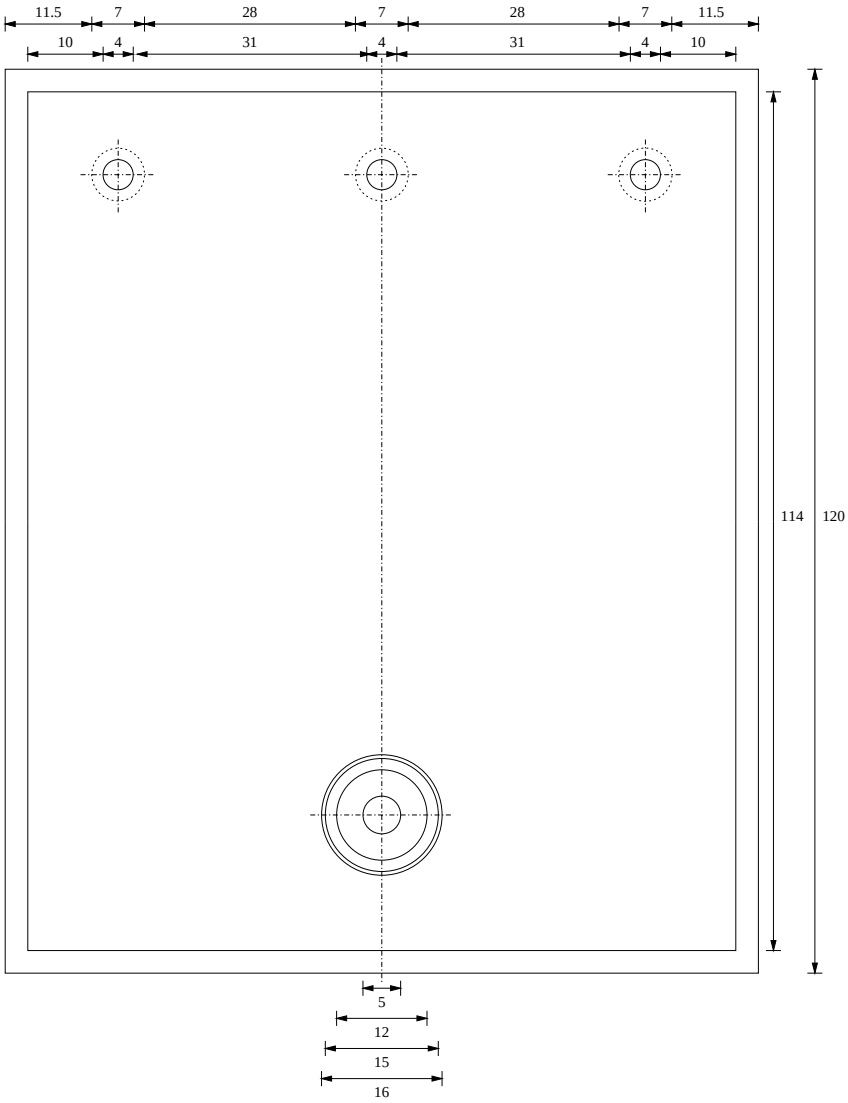


Figure C.12: Gimbal base plate with pedestal for the gimbal rings and cross assembly, (view 1).

The gimbal base plate, figure C.12 and C.13, has a raised pedestal at one end, in which the base plate low friction polymer bearing fits, and three 4*mm* holes at the other end which hold the backplate of the gimbal assembly in place.

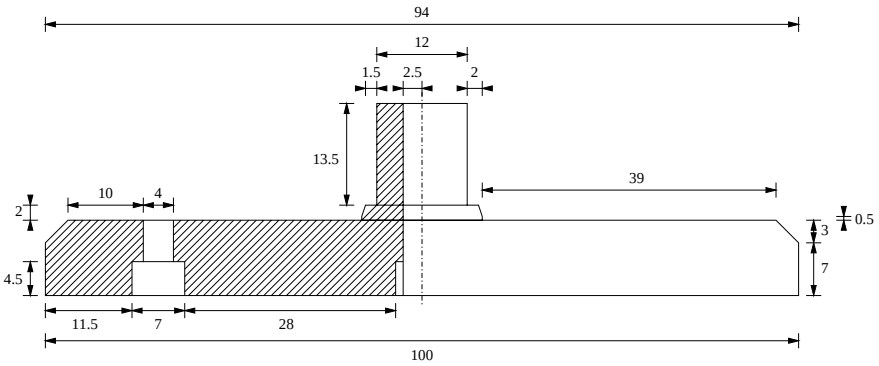


Figure C.13: Gimbal base plate with pedestal for the gimbal rings and cross assembly, (view 2).

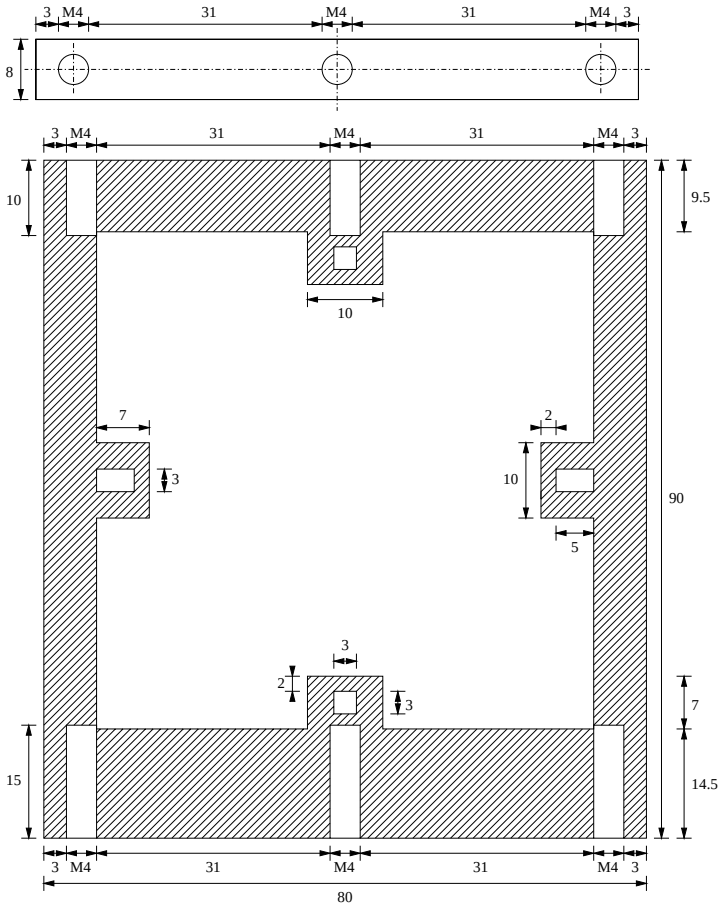


Figure C.14: Gimbal back plate with taps for assembling the base and top plate.

The gimbal back plate C.14 is mounted by three *M*4 screws to one end of the gimbal base plate. The machined area of the back plate provides a space through which rods can pass, connecting the gimbal with the solenoids.

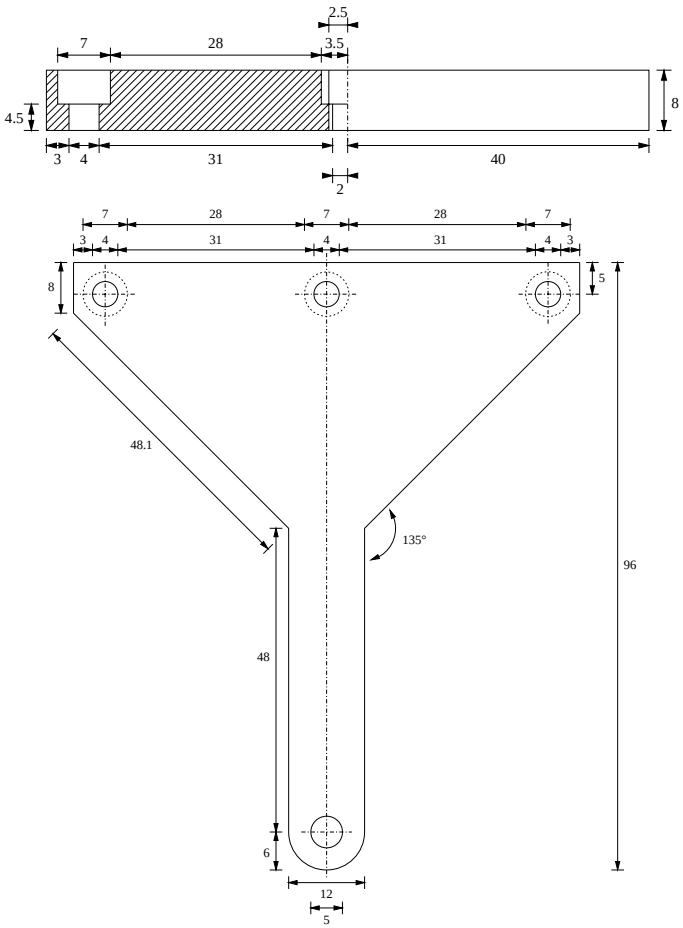


Figure C.15: Gimbal top plate.

The gimbal top plate, figure C.15, has a 5*mm* hole at one end, in which the top plate low friction polymer bearing fits, and three 4*mm* holes for *M*4 screws at the other end.

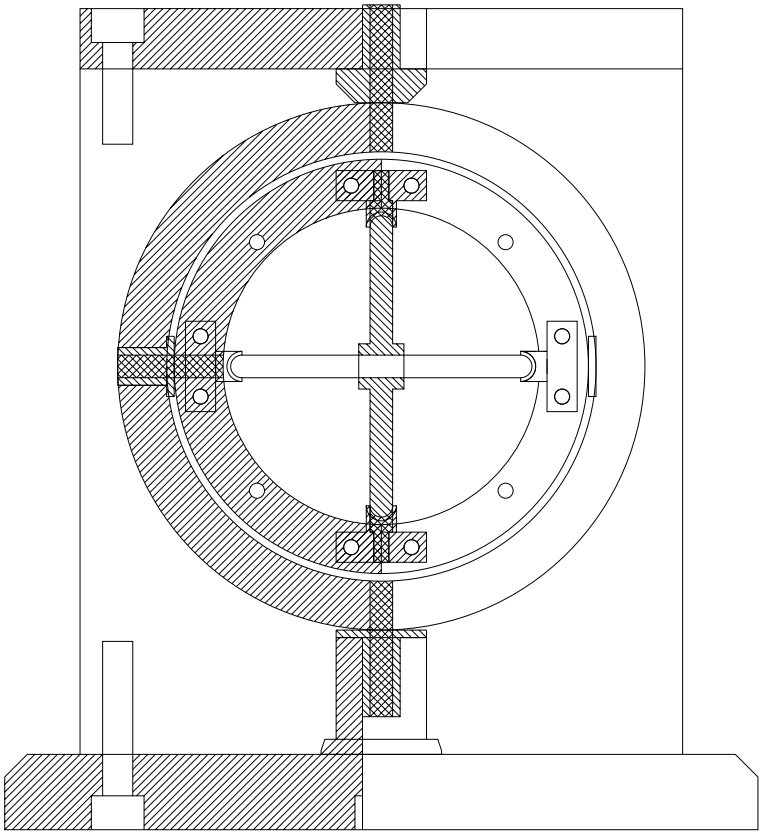


Figure C.16: Complete gimbal assembly.

The gimbal assembly in figure C.16 shows all components except for the screws and gimbal cross spacers. Larger components (gimbal frame assembly), and those subjected to high accelerations (gimbal rings) are manufactured out of aluminium, to increase performance of the platform and also reduce the overall weight of the completed system. Smaller components (gimbal cross assembly) are manufactured out of steel and brass, to withstand the torque of fully activated solenoids.

C.4 Images

The following images demonstrate the completed and integrated gimbal assembly.

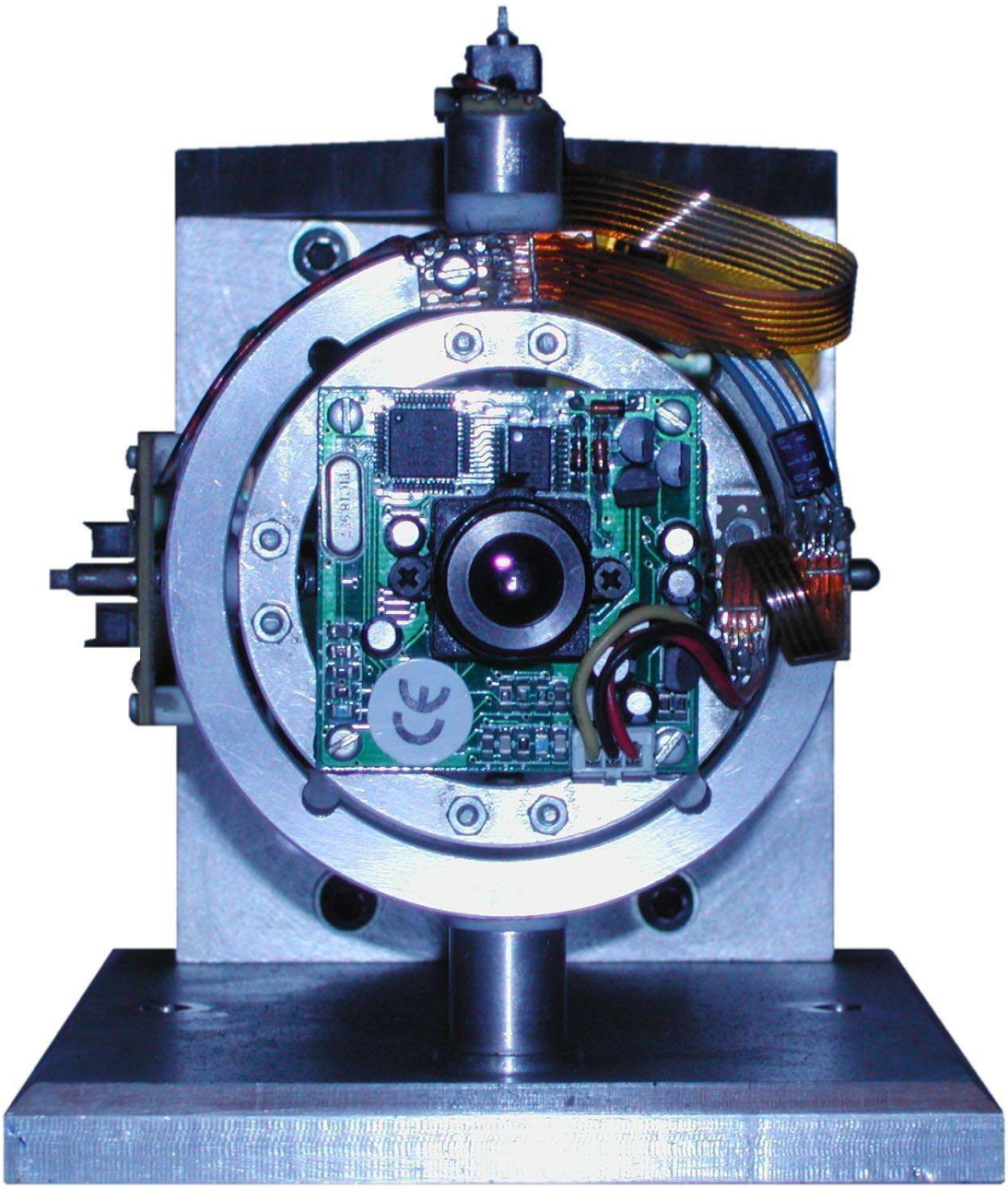


Figure C.17: Front view of the completed gimbal assembly with mounted camera, solenoid activation unit and damping unit. The wiring and a form of rough gimbal position feedback is also implemented (two potentiometers can be mounted on top of the gimbal top plate and the extreme left side of the outer gimbal ring).

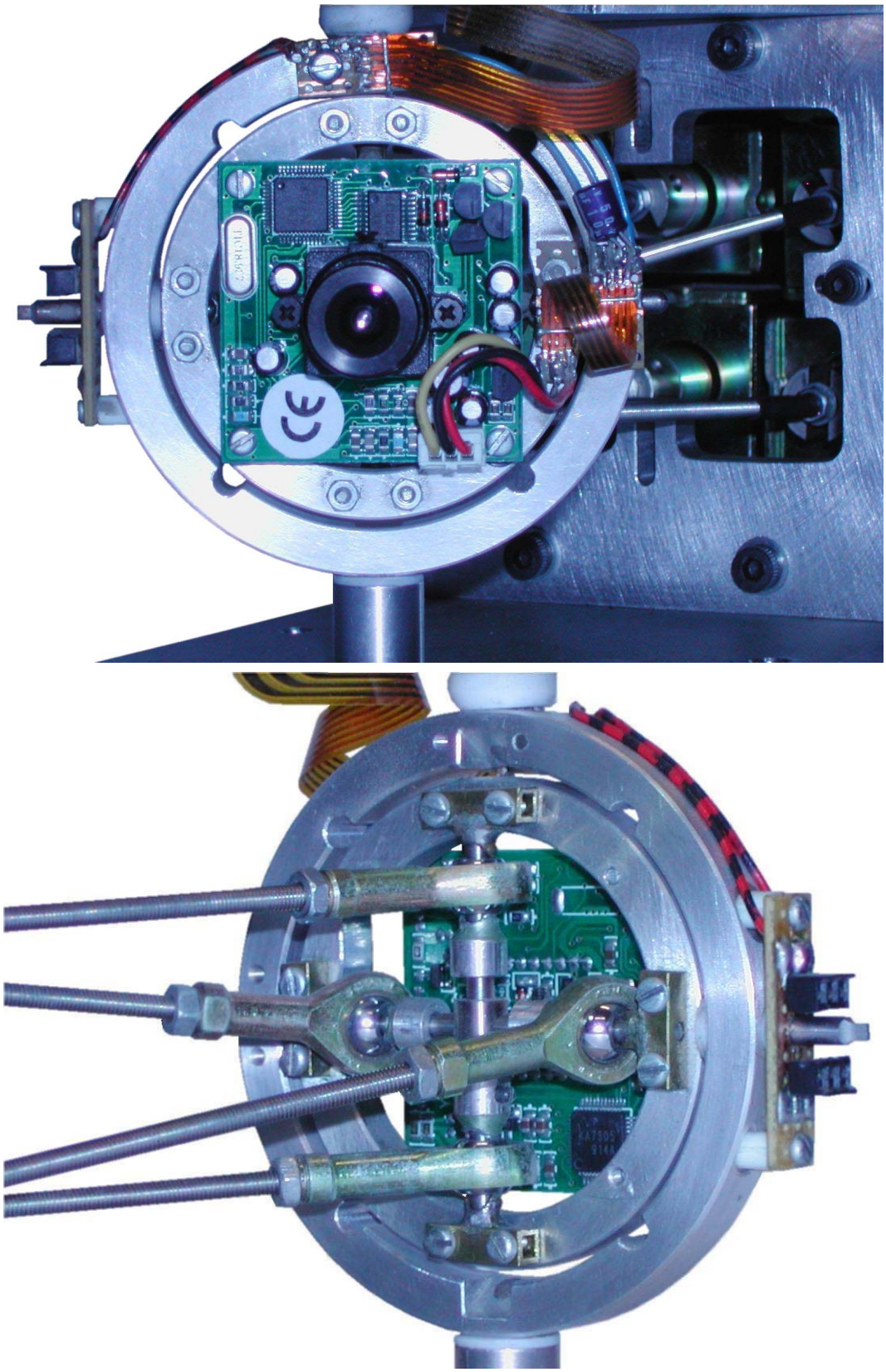


Figure C.18: Closeup view of gimbal ring and gimbal cross assembly with mounted camera, rod-end bearings and rods. The mechanism for rough gimbal tilt position feedback is also visible (a potentiometer can be mounted on the extreme side of the outer gimbal ring).

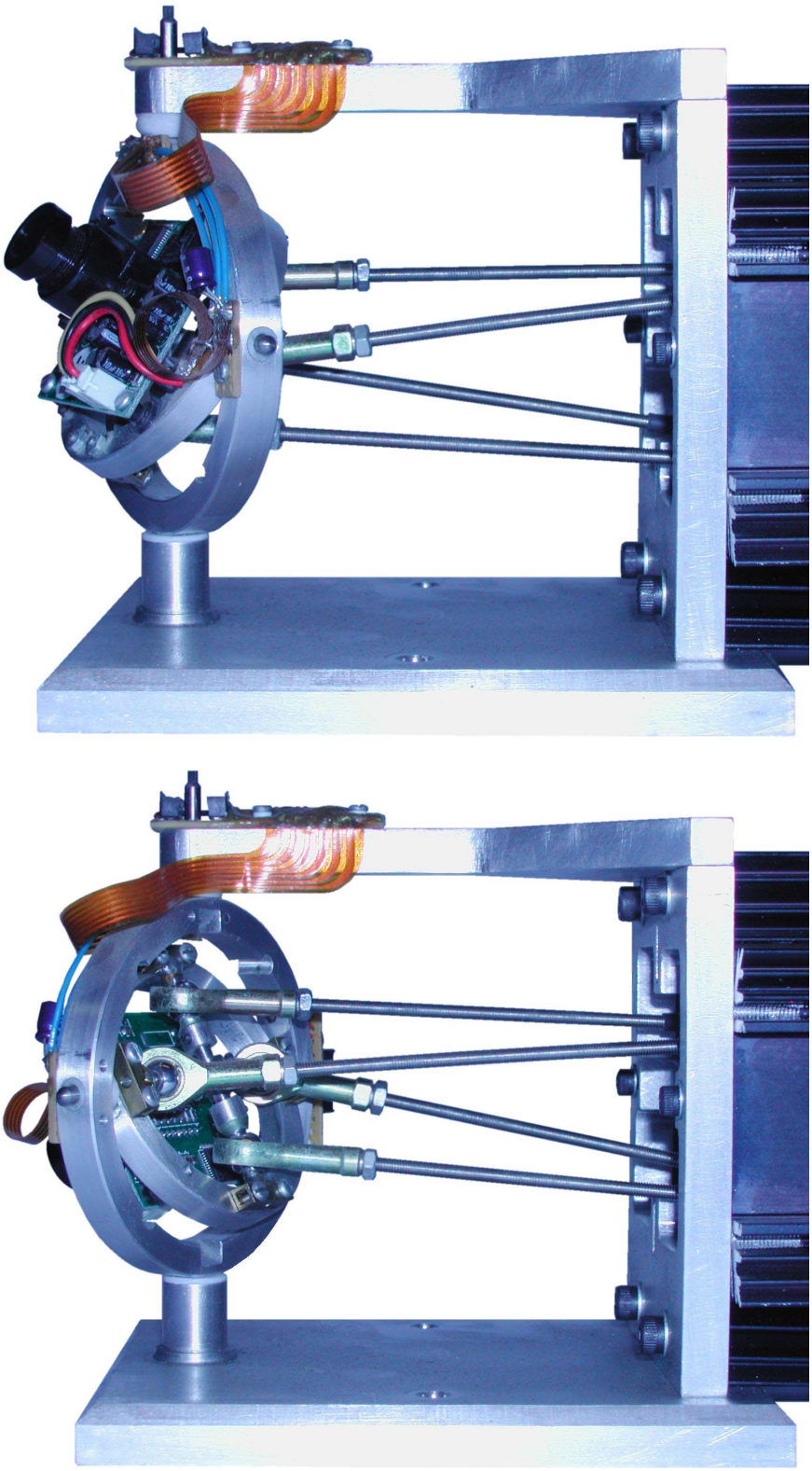


Figure C.19: Side view of gimbal ring and gimbal cross assembly with mounted camera, rod-end bearings and rods. The mechanism for rough gimbal pan position feedback is also visible (a potentiometer can be mounted on top of the gimbal top plate). The gimbal is positioned in two extreme pan and tilt positions.

Appendix D

System Assembly

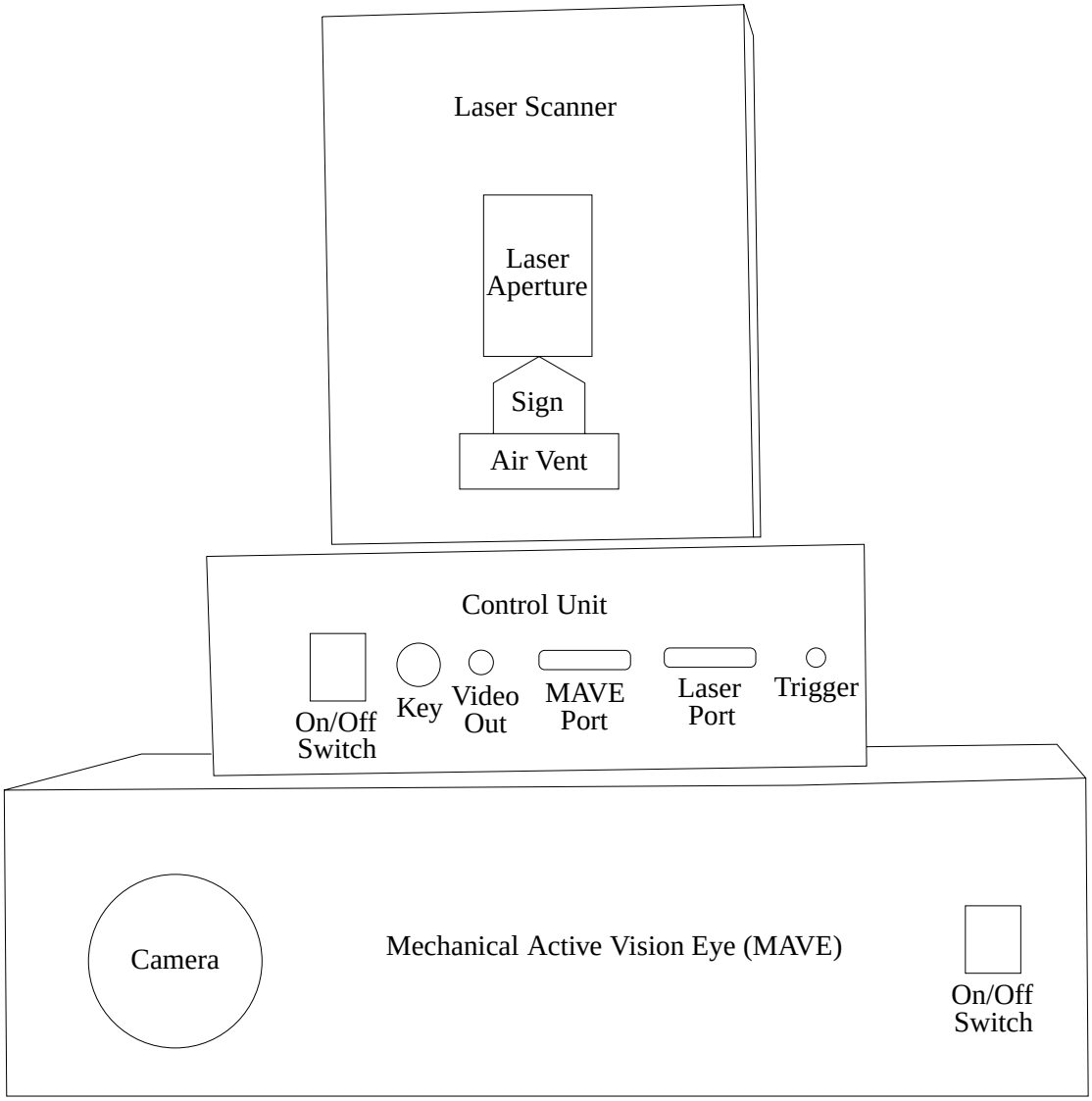
The robotic components, designed specifically for this research, are housed in three separate cases, figure D.1. This appendix takes a high level view of these three cases and identifies the location and arrangement of significant components. Appendices B and C provide more detailed information on the assembly of the mechanics and the interaction of electrical components.

Figure D.2 shows a front view of the laser scanner with the stepper motors for shutter control and vertical target movement clearly visible. The location of the laser source, the light barrier for shutter positioning and the photo transistors for horizontal and vertical mirror calibration are also highlighted. Figure D.3 shows a rear view of the laser scanner, with all three stepper motors visible. The location of the laser source, the light barrier and the photo transistors are also highlighted.

Figure D.4 exposes the main interface and control electronics. The computer interface board with its own power supply and three stepper motor controllers are the central components. The large power supply unit supplies the stepper motor controllers and the components of the laser scanner.

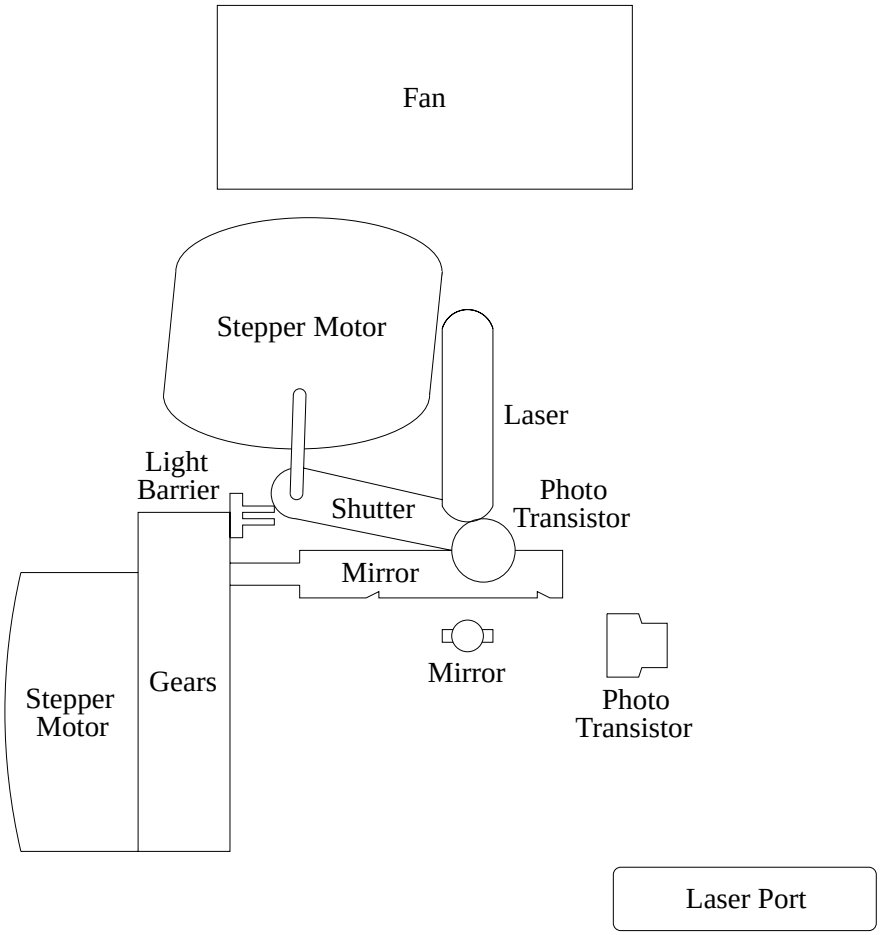
Figure D.5 shows the internal components of the mechanical Monocular Active Vision Eye (MAVE). The power supply and regulation components are in the top half of the image. The pulse width modulators that drive the solenoids are arranged in the middle of the image, and the mechanics and camera of the MAVE are arranged at the bottom of the image.

The images in this appendix are not uniformly scaled. Some images also appear slight distortions, due to the camera set up at the time of picture acquisition.


Laser Scanner
Laser Aperture
Sign
Air Vent
Control Unit
On/Off Switch
Key
Video Out
MAVE Port
Laser Port
Trigger
Camera
Mechanical Active Vision Eye (MAVE)
On/Off Switch

D.1 All Units

Figure D.1: The three encased experimental hardware units. The laser scanner, the control unit and the mechanical MAVE.


Fan
Stepper Motor
Laser
Light
Barrier
Shutter
Photo
Transistor
Mirror
Mirror
Photo
Transistor
Stepper
Motor
Gears
Laser Port

D.2 Laser Scanner

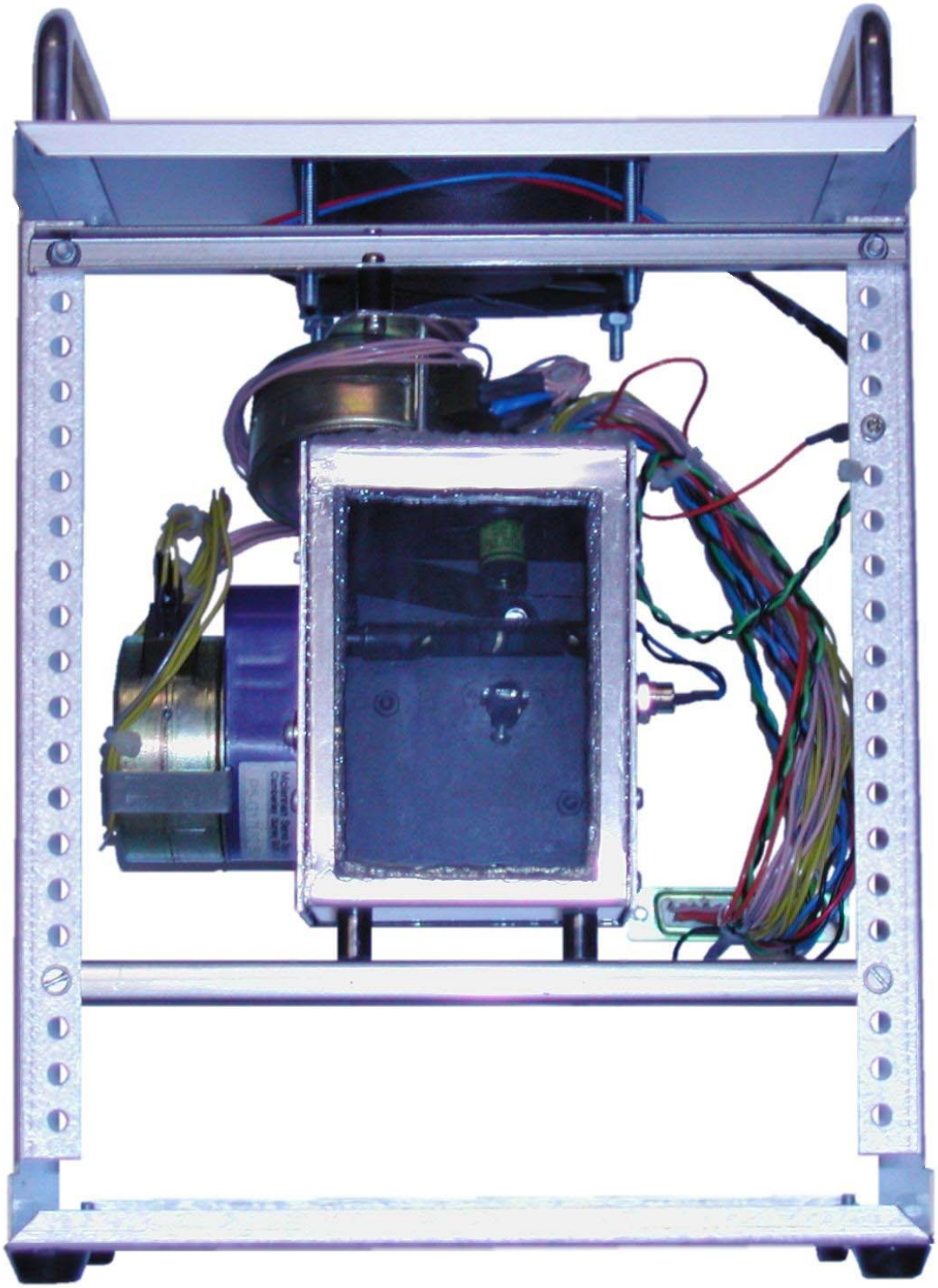
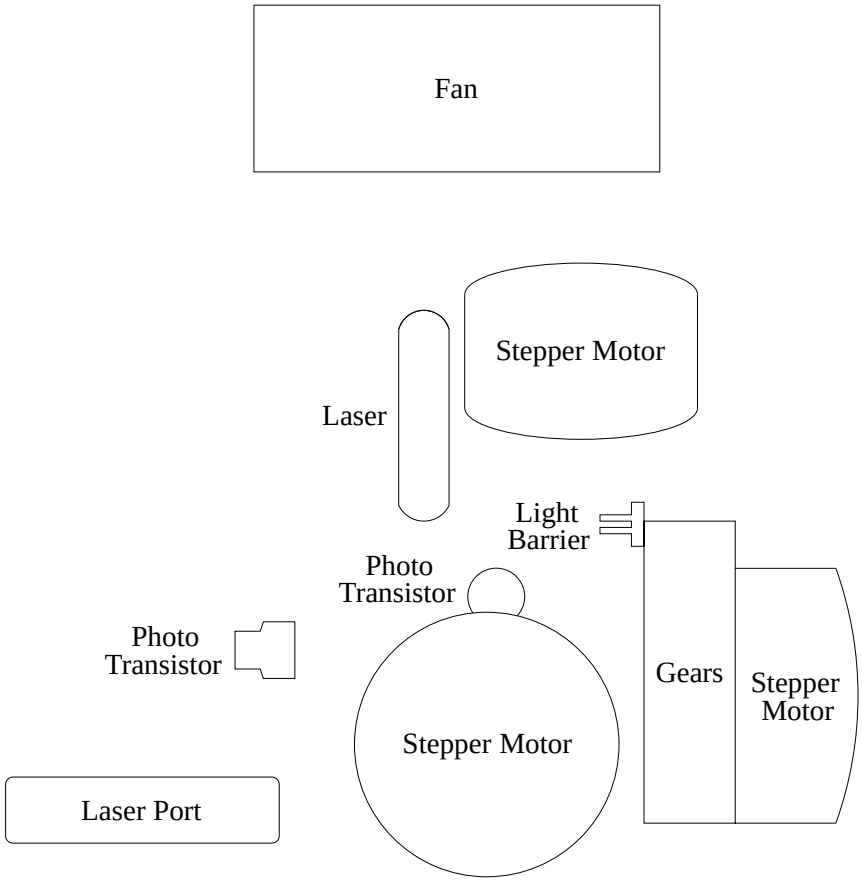


Figure D.2: Front view of the laser scanner with the case removed.


Fan
Stepper Motor
Laser
Light
Barrier
Photo
Transistor
Photo
Transistor
Gears
Stepper
Motor
Stepper Motor
Laser Port

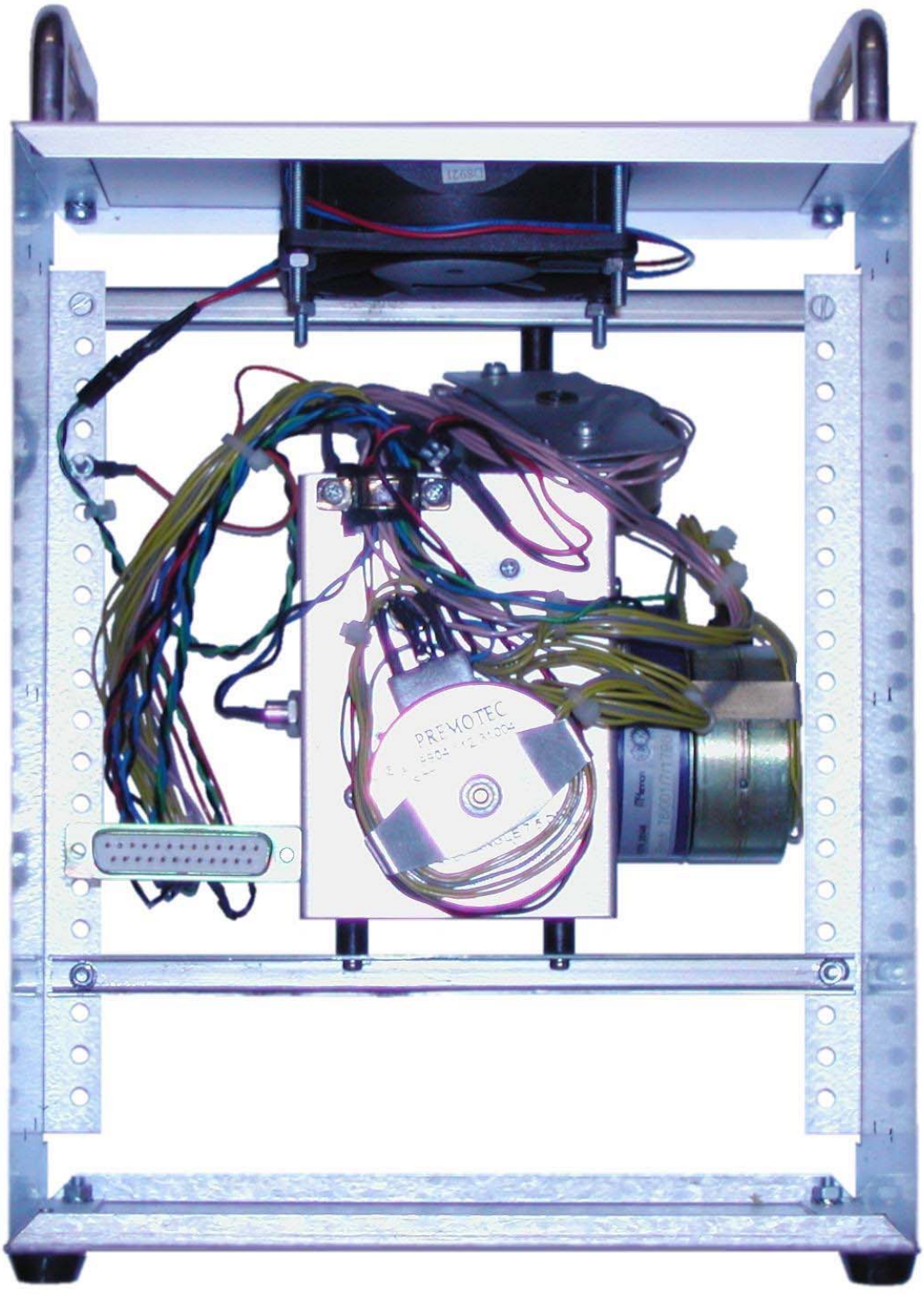
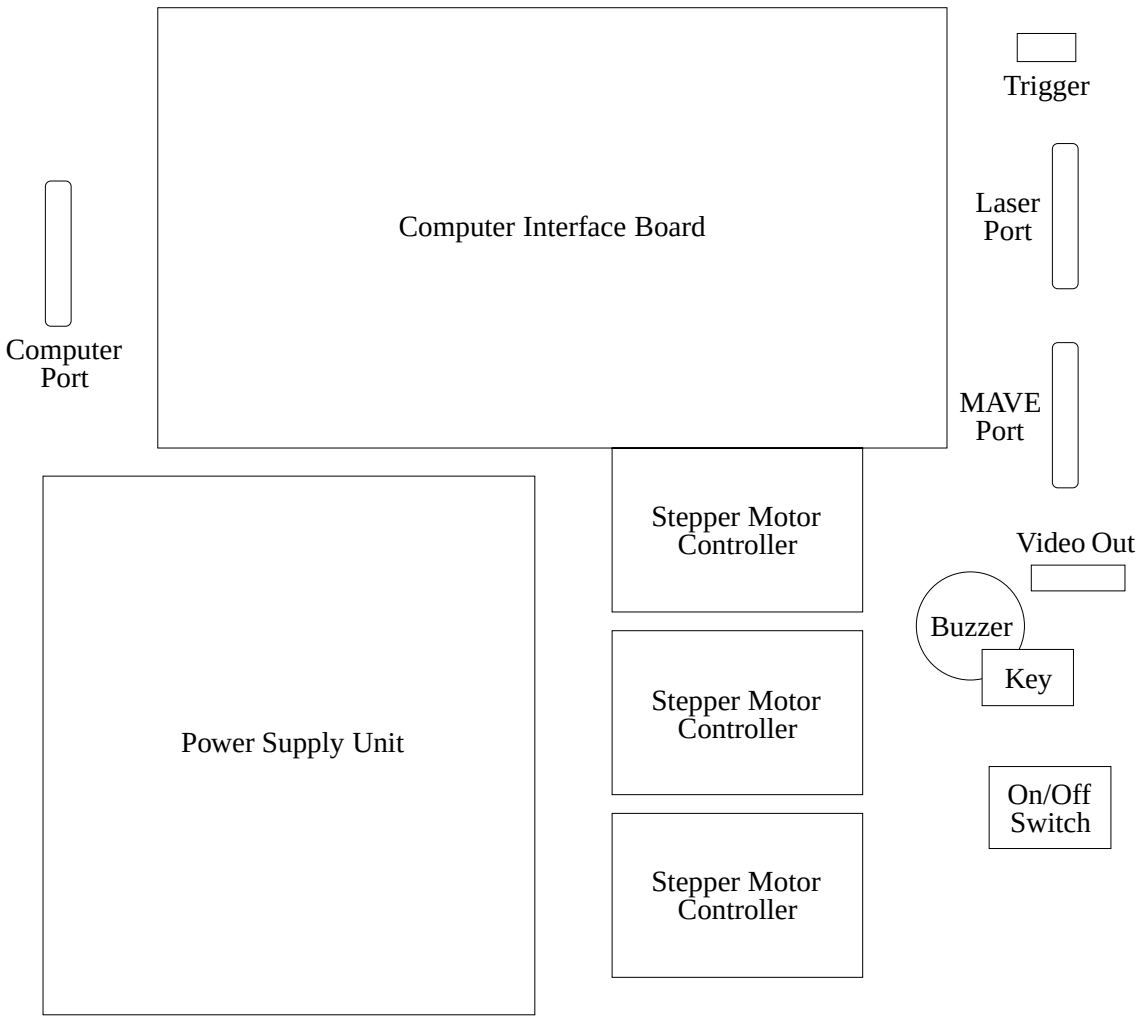


Figure D.3: Rear view of the laser scanner with the case removed.


Computer Interface Board
Trigger
Laser Port
MAVE Port
Computer Port
Video Out
Buzzer
Key
Stepper Motor Controller
Stepper Motor Controller
Stepper Motor Controller
Power Supply Unit
On/Off Switch

D.3 Control Unit

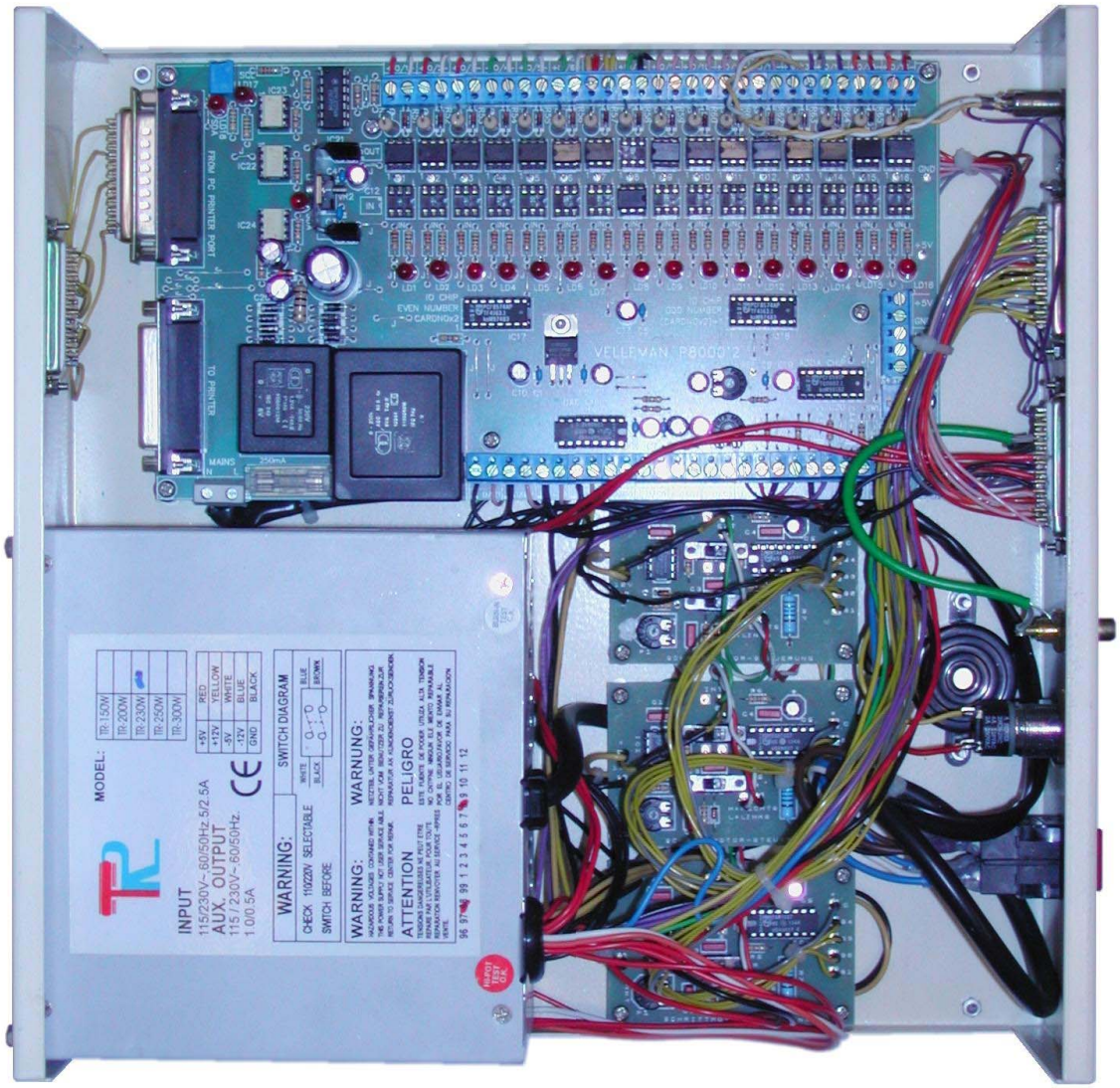
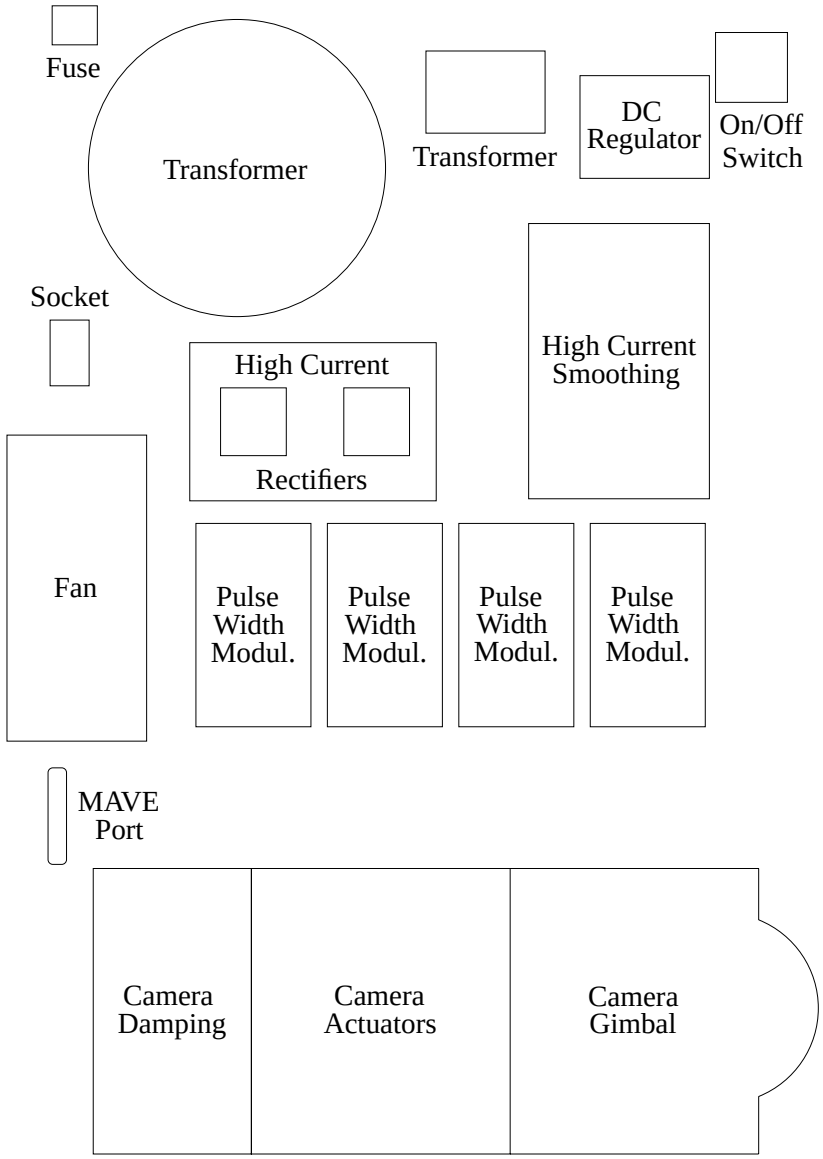


Figure D.4: Top view of the control unit with the case removed.


Fuse
Transformer
Transformer
DC Regulator
On/Off Switch
Socket
High Current
Rectifiers
High Current Smoothing
Fan
Pulse Width Modul.
Pulse Width Modul.
Pulse Width Modul.
Pulse Width Modul.
MAVE Port
Camera Damping
Camera Actuators
Camera Gimbal

D.4 Monocular Active Vision Eye

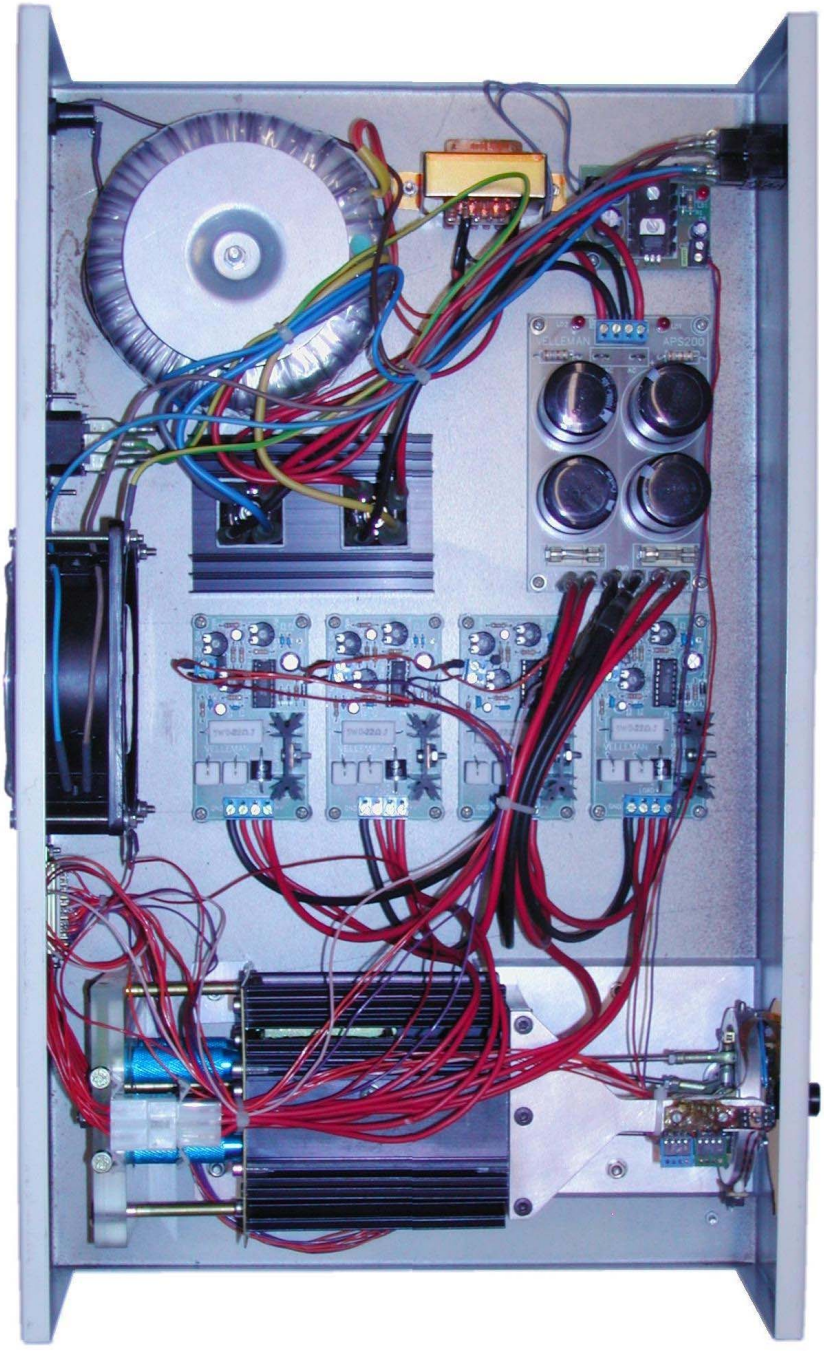


Figure D.5: Top view of the mechanical MAVE with the case removed.

Appendix E

Acronyms

AC	Alternating Current
ADP	Adenosine Di Phosphate
AI	Artificial Intelligence
ATP	Adenosine Tri Phosphate
CCD	Charge Coupled Device
COGS	School of Cognitive and Computing Sciences
CPU	Central Processing Unit
DAI	Department of Artificial Intelligence
DC	Direct Current
DMA	Direct Memory Access
DOS	Disc Operating System
EMF	Electro Magnetic Field
FSM	Finite State Machine
I^2C	Inter-IC bus
IC	Integrated Circuit
LED	Light Emitting Diode
LSL	Langsame Störsichere Logikfamilie
LVR	Laser Videodisc Recorder
mA	Milli Ampere
MAVE	Monocular Active Vision Eye
MB	Mega Byte
PSU	Power Supply Unit
PW	Pulse Width
PWM	Pulse Width Modulation
RAM	Random Access Memory
SMC	Stepper Motor Controller
SVGA	Super Video Graphics Accelerator
VDC	Volt DC
VOR	Vestibulo-Ocular Reflex

Bibliography

[1] Safety in Universities Notes of Guidance Part 2:1 Lasers. University of Sheffield Printing Unit, The Committee of Vice-Chancellors and Principals of the Universities of the United Kingdom, 29 Tavistock Square, London, WC1H 9EZ, October 1992.

[2] The University of Sussex Safety Committee Local Rules for Work with Lasers, February 1992. SSC-56-2.

[3] Y. Aloimonos. *Active Perception*, chapter Active Vision Revised, pages 1–18. Lawrence Erlbaum Associates Inc., 1993.

[4] Y. Aloimonos, I. Weiss, and A. Bandyopadhyay. Active Vision. In *First IEEE Conference on Computer Vision, London*, pages 35–54, June 1987.

[5] M. Alpern. Anatomy of Eye Movements. In H. Davson, editor, *The Eye*, volume 3, pages 27–64. Academic Press, 2nd edition, 1969.

[6] M. Alpern. Kinematics of the Eye. In H. Davson, editor, *The Eye*, volume 3, pages 13–25. Academic Press, 2nd edition, 1969.

[7] M. Alpern. Physiological Characteristics of the Extraocular Muscles. In H. Davson, editor, *The Eye*, volume 3, pages 175–202. Academic Press, 2nd edition, 1969.

[8] AP-15A & AP-15D Pan & Tilt Heads. Internet. **http://www.aritech.com/Products/cctv.htm** 01.04.2002.

[9] D. B. Arnold and D. A. Robinson. A Neural Network Model of the Vestibulo-Ocular Reflex using a Local Synaptic Learning Rule. *Philosophical Transactions of The Royal Society of London Series B - Biological Sciences*, 337(1281):327–330, September 1992.

[10] A. T. Bahill and L. Stark. Overlapping Saccades and Glissades are Produced by Fatigue in the Saccadic Eye Movement System. *Experimental Neurology*, 48:95–106, February 1975.

[11] A. T. Bahill and L. Stark. The Trajectories of Saccadic Eye Movements. *Scientific American*, 240(1):85–93, January 1979.

[12] S.-W. Ban, J.-K. Cho, S.-K. Jung, and M. Lee. Active Vision System Based on Human Eye Saccadic Movement. *IEICE Transactions of Fundamentals of Electronics Communication and Computer Sciences*, E83-A(6):1066–1074, June 2000.

[13] R. Beal and T. Jackson. *Neural Computing: An Introduction*. Institute of Physics Publishing LTD, 1994.

[14] W. Becker. Saccades. In R. H. S. Carpenter, editor, *Vision and Visual Dysfunction - Eye Movements*, volume 8, pages 95–137. Macmillan Press LTD, 1991.

[15] W. Becker and R. Jürgens. Human Oblique Saccades: Quantitative Analysis of the Relation between Horizontal and Vertical Components. *Vision Research*, 30(6):893–920, 1990.

[16] L. Berthouze, S. Rougeaux, F. Chavand, and Y. Kuniyoshi. Calibration of a Foveated Wide Angle Lens on an Active Vision Head. In *IEEE/PAMI Computer Vision and Pattern Recognition, San Francisco, USA*, June 1996.

[17] G. Bielig-Schulz and C. Schulz. *3D-Graphik in PASCAL*. B. G. Teubner, 1987.

[18] A. Blake and A. Yuille. *Active Vision*. MIT Press, 1993.

[19] K. J. Bradshaw, P. F. McLauchlan, I. D. Reid, and D. W. Murray. Saccade and Pursuit on an Active Head/Eye Platform. *Image and Vision Computing*, 12(3):155–163, April 1994.

[20] J. Brodkey and L. Stark. New Direct Evidence against Intermittency of Sampling in Human Smooth Pursuit Eye Movements. *Nature*, 218:273–275, April 1968.

[21] A. Brooks, G. Dickins, A. Zelinsky, J. Kieffer, and S. Abdallah. A High-Performance Camera Platform for Real-Time Active Vision. In *Proceedings of the First International Conference on Field and Service Robotics*, Robotic Systems Laboratory, Department of Engineering (FEIT), Australian National University, Canberra, ACT 0200 Australia, 1997.

[22] C. Brown, D. Coombs, and J. Soong. Real-Time Smooth Pursuit Tracking. In A. Blake and A. Yuille, editors, *Active Vision*, pages 123–136. MIT Press, 1993.

[23] R. M. Burde. Control of Eye Movements. In R. A. Moses, editor, *Adler's Physiology of the Eye - Clinical Application*, pages 122–165. The C. V. Mosby Company, 7th edition, 1981.

[24] C. Burdess. Saccadic Eye Movements: Oculomotor Control in the Superior Colliculus. Internet. **http://www.dog.net.uk/saccades/** 01.04.2002.

[25] H. Buxton. Visual Interpretation and Understanding. Technical Report CSRP 452, School of Cognitive and Computing Sciences, University of Sussex, Brighton BN1 9QH, UK, January 1997.

[26] G. H. Byford. Sight, Hearing and Balance. In D. A. Linkens, editor, *Biological Systems, Modelling and Control: IEE Control Engineering Series 11*, pages 110–137. Peter Peregrinus LTD, 1979.

[27] M. B. Carpenter. Central Oculomotor Pathways. In P. Bach y Rita and C. C. Collins, editors, *The Control of Eye Movements*, pages 67–103, Smith-Kettlewell Institute of Visual Sciences, University of the Pacific Graduate School of Medical Sciences, San Francisco, California, 1971. Academic Press Inc.

[28] R. H. Carpenter. Eye Movements. Internet. **http://www.cai.ac.uk/subj/medicine/oculo.html** 22.11.1998.

[29] H. I. Christensen. The AUC Robot Camera Head. Internal Report, October 1995.

[30] H. I. Christensen, J. Horstmann, and T. Rasmussen. A Control Theoretical Approach to Active Vision. Technical report, Laboratory of Image Analysis, Institute of Electronic Systems, Aalborg University, Fr. Bajers Vej, DK-9220 Aalborg East, 1995.

[31] J. J. Clark and N. J. Ferrier. Modal Control of an Attentive Vision System. In *Second IEEE Conference on Computer Vision, Tarpon Spring, Florida*, pages 514–523, December 1988.

[32] J. J. Clark and N. J. Ferrier. Attentive Visual Servoing. In A. Blake and A. Yuille, editors, *Active Vision*, pages 137–154. MIT Press, 1993.

[33] H. Collewijn, R. M. Steinman, C. J. Erkelens, and D. Regan. Binocular Fusion, Stereopsis and Stereoacuity with a Moving Head. In D. Regan, editor, *Vision and Visual Dysfunction - Binocular Vision*, volume 9, pages 121–136. Macmillan Press LTD, 1991.

[34] C. C. Collins. Orbital Mechanics. In P. Bach y Rita and C. C. Collins, editors, *The Control of Eye Movements*, pages 283–325, Smith-Kettlewell Institute of Visual Sciences, University of the Pacific Graduate School of Medical Sciences, San Francisco, California, 1971. Academic Press Inc.

[35] C. C. Collins. The Human Oculomotor Control System. In G. Lennerstrand and P. Bach y Rita, editors, *Basic Mechanisms of Ocular Motility and their Clinical Implications*, pages 145–180. Pergamon Press, June 1974.

[36] C. Currie. Visual Cognition: The Role of Eye Movements in Real World Scene Perception. Internet. **http://kahuna.cogsci.uiuc.edu/ipl/cog/scene/sw_intro1.html** 22.11.1998.

[37] H. Davson, editor. *The Eye*, volume 3. Academic Press Inc., 2nd edition, 1969.

[38] F. C. Donders. Die Bewegungen des Auges, veranschaulicht durch das Phaenophthalmotrop. *Archives of Ophthalmology*, 16(1):154–175, 1870.

[39] L. Festinger. Eye Movements and Perception. In P. Bach y Rita and C. C. Collins, editors, *The Control of Eye Movements*, pages 259–273, Smith-Kettlewell Institute of Visual Sciences, University of the Pacific Graduate School of Medical Sciences, San Francisco, California, 1971. Academic Press Inc.

[40] M. A. Fischler and O. Firschein. *Intelligence the Eye, the Brain and the Computer*. Addison-Wesley, 1987.

[41] J. M. Foley. Binocular Space Perception. In D. Regan, editor, *Vision and Visual Dysfunction - Binocular Vision*, volume 9, pages 75–92. Macmillan Press LTD, 1991.

[42] N. Franceschini, J. Pichon, and C. Blanes. From Insect Vision to Robot Vision. *Philosophical Transactions of The Royal Society of London Series B - Biological Sciences*, 337(1281):283–294, September 1992.

[43] A. F. Fuchs. The Saccadic System. In P. Bach y Rita and C. C. Collins, editors, *The Control of Eye Movements*, pages 343–362, Smith-Kettlewell Institute of Visual Sciences, University of the Pacific Graduate School of Medical Sciences, San Francisco, California, 1971. Academic Press Inc.

[44] G. Gancarz and S. Grossberg. A Neural Model of Saccadic Eye Movement Control explains Task-Specific Adaptation. Technical Report CAS/CNS-TR-98-024, Department of Cognitive and Neural Systems and Centre for Adaptive Systems, Boston University, 677 Beacon Street, Boston, MA 02215, July 1999.

[45] H. M. Gomes, R. B. Fisher, and J. Hallam. A Retina-like Image Representation of Primal Sketch Features Extracted using a Neural Network Approach. Technical Report 1115, Department of Artificial Intelligence, University of Edinburgh, 5 Forrest Hill, Edinburgh, EH1 2QL, Scotland, 1998.

[46] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Addison Wesley, 1993.

[47] C. Gosselin, F. Caron, C. Cloutier, P. Dupont, M. Gagné, M. Jean, M. Lambert, E. Lavoie, P. Le-Huu, M. Leblond, S. Lemieux, L. Perreault, N. Pouliot, R. Ricard, J. Sefrioui, B. M. Saint-Onge, C. Vaillancourt, and J. Wang. Kinematics and Dynamics of Parallel Mechanisms and Manipulators. Internet. **http://wwwrobot.gmc.ulaval.ca/ANG/rob_porall_an. html** 01.04.2002.

[48] C. M. Gosselin, É. St-Pierre, and M. Gagné. On the Development of the Agile Eye: Mechanical Design, Control Issues and Experimentation. *IEEE Robotics and Automation Society Magazine*, 3(4), December 1996.

[49] P. Gouras. Precortical Physiology of Colour Vision. In P. Gouras, editor, *Vision and Visual Dysfunction - The Perception of Colour*, volume 6, pages 163–178. Macmillan Press LTD, 1991.

[50] P. Gouras. The History of Colour Vision. In P. Gouras, editor, *Vision and Visual Dysfunction - The Perception of Colour*, volume 6, pages 1–9. Macmillan Press LTD, 1991.

[51] R. Granit. The Probable Role of Muscle Spindles and Tendon Organs in Eye Movement Control. In P. Bach y Rita and C. C. Collins, editors, *The Control of Eye Movements*, pages 3–5, Smith-Kettlewell Institute of Visual Sciences, University of the Pacific Graduate School of Medical Sciences, San Francisco, California, 1971. Academic Press Inc.

[52] M. P. Groover, M. Weiss, R. N. Nagel, and N. G. Odrey. *Industrial Robotics Technology, Programming and Applications*. McGraw-Hill International Editions, 1986.

[53] L. Hainline. Normal Lifespan Developmental Changes in Saccadic and Pursuit Eye Movements. In C. W. Johnston and F. J. Pirozzolo, editors, *Neuropsychology of Eye Movements*, pages 31–64. Lawrence Erlbaum Associates Inc., 1988.

[54] J. Hertz, A. Krogh, and R. G. Palmer. *Introduction to the Theory of Neural Computation*. Addison Wesley, 1991.

[55] hihihi. I2C-Bus Libraries for Linux. Internet. **http://home.wanadoo.nl/hihihi/** 01.04.2002.

[56] G. A. Horridge. What can Engineers Learn from Insect Vision? *Philosophical Transactions of The Royal Society of London Series B - Biological Sciences*, 337(1281):271–282, September 1992.

[57] E. Hultman, H. Sjoholm, K. Sahlin, and L. Edstrom. Glycolytic and Oxidative Energy Metabolism and Contraction Characteristics of Intact Human Muscle. In Ruth Porter and Julie Whelan, editors, *Human Muscle Fatigue: Physiological Mechanisms*, pages 19–40. Pitman Medical, London, May 1981. Ciba Foundation symposium 82.

[58] F. H. Körner. Non-Visual Control of Human Saccadic Eye Movements. In G. Lennerstrand and P. Bach y Rita, editors, *Basic Mechanisms of Ocular Motility and their Clinical Implications*, pages 565–569. Pergamon Press, June 1974.

[59] P. C. Kronfeld. The Gross Anatomy and Embryology of the Eye. In H. Davson, editor, *The Eye*, volume 1, pages 1–3. Academic Press, 2nd edition, 1969.

[60] E. P. Krotkov. An Agile Stereo Camera System. In Ramesh Jain, editor, *Active Computer Vision by Cooperative Focus and Stereo*, pages 7–18. Springer-Verlag, 1989.

[61] Y. Kuniyoshi, N. Kita, S. Rougeaux, and T. Suehiro. Active Stereo Vision System with Foveated Wide Angle Lenses. In *Asian Conference on Computer Vision, Singapore*, 1995.

[62] Y. Kuniyoshi, N. Kita, K. Sugimoto, S. Nakamura, and T. Suehiro. A Foveated Wide Angle Lens for Active Vision. In *IEEE International Conference on Robotics and Automation*, 1995.

[63] P. Lancaster and K. Salkauskas. *Curve and Surface Fitting: An Introduction*. Academic Press, 1986.

[64] M. F. Land. Motion and Vision: Why Animals Move their Eyes. *Journal of Comparative Physiology*, 185:341–352, April 1999.

[65] M. F. Land and R. D. Fernald. The Evolution of Eyes. *Annual Review of Neuroscience*, 15:1–29, 1992.

[66] M. F. Land and S. Furneaux. The Knowledge Base of the Oculomotor System. *Philosophical Transactions of The Royal Society of London Series B - Biological Sciences*, 352(1358):1231–1239, August 1997.

[67] M. F. Land and J. Horwood. Which Parts of the Road Guide Steering? *Nature*, 377:339–340, September 1995.

[68] M. F. Land and J. Horwood. The Relations Between Head and Eye Movements During Driving. *VISION IN VEHICLES - V*, pages 153–160, October 1996.

[69] M. F. Land and D. N. Lee. Where we Look when we Steer. *Nature*, 369:742–744, June 1994.

[70] M. F. Land and D.-E. Nilsson. *Animal Eyes*. Oxford University Press, 2002.

[71] V. P. Laurutis and D. A. Robinson. The Vestibulo-Ocular Reflex During Human Saccadic Eye Movements. *Journal of Physiology-London*, 373:209–233, April 1986.

[72] E. H. Leaton. The Scalar Product. In D. Wheeler, editor, *Vectors*, number 3, pages 79–87. George Allen and Unwin LTD, 1968.

[73] D. N. Lee. Body-Environment Coupling. In U. Neisser, editor, *The Perceived Self - Ecological and Interpersonal Sources of Self-Knowledge*, pages 43–67. Cambridge University Press, 1993.

[74] B. C. Madden and U. M. Cahn von Seelen. PennEye a Binocular Active Vision System. Technical report, Department of Computer and Information Science, University of Pennsylvania, 3401 Walnut Street, Room 301C, Philadelphia, PA 19104, USA, December 1995.

[75] R. Manzotti, R. Tiso, E. Grosso, and G. Sandini. Primary Ocular Movements Revisited. Technical Report LIRA-TR 7/94, LIRA-Lab DIST University of Genova, Via Opera Pia 13, 16145 Genova, Italy, November 1994.

[76] M. J. Marjanović. Our Wonderful Cerebellum - An Insight into the Organ from Tomorrow. Internet, 1993. **http://www.ai.mit.edu/people/maddog/maddog.html** 01.04.2002.

[77] D. Marr. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. W. H. Freeman and Company, 1982.

[78] P. F. McLauchlan, I. D. Reid, S. M. Fairley, and D. W. Murray. The Pipe-Group Architecture for Real-Time Active Vision. *Real-Time Imaging*, 3(5):319–330, October 1997.

[79] H. T. Milhorn. Human Eye Tracking of Aperiodic Functions. In *The Application of Control Theory to Physiological Systems*, pages 255–282. W. B. Saunders Company, 1966.

[80] J. E. Miller. Aging Changes in Extraocular Muscle. In G. Lennerstrand and P. Bach y Rita, editors, *Basic Mechanisms of Ocular Motility and their Clinical Implications*, pages 47–61. Pergamon Press, June 1974.

[81] G. P. Moore. Cramming More Components onto Integrated Circuits. *Electronics*, 38(1):114–117, April 1965.

[82] D. C. Murdoch. *Analytical Geometry with an Introduction to Vectors and Matrices*, chapter Planes, Lines and Spheres in ε_3. John Wiley and Sons Inc., 1966.

[83] D. W. Murray, K. J. Bradshaw, P. F. McLauchlan, I. D. Reid, and P. M. Sharkey. Driving Saccade to Pursuit using Image Motion. *International Journal of Computer Vision*, 16(3):205–228, November 1995.

[84] D. W. Murray, F. Du, P. F. McLauchlan, I. D. Reid, P. M. Sharkey, and M. Brady. Design of Stereo Heads. In A. Blake and A. Yuille, editors, *Active Vision*, pages 155–172. MIT Press, 1993.

[85] D. W. Murray, P. F. McLauchlan, I. D. Reid, and P. M. Sharkey. Reactions to Peripheral Image Motion using a Head/Eye Platform. *International Conference on Computer Vision*, pages 403–411, 1993.

[86] F. Panerai, G. Metta, and G. Sandini. Adaptive Image Stabilization: A Need for Vision-based Active Robotic Agents. In *International Conference on Simulation of Adaptive Behaviour, Paris, France*, 2000.

[87] F. Panerai and G. Sandini. Oculo-Motor Stabilization Reflexes: Integration of Inertial and Visual Information. *Neural Networks*, 11(7/8):1191–1204, October/November 1998.

[88] F. Pardo and E. Martinuzzi. Hardware Environment for a Retinal CCD Visual Sensor. In *EU-HCM SMART Workshop: Semi-autonomous Monitoring and Robotics Technologies, Ispra, Italy*, April 1994.

[89] L. Peachey. The Structure of the Extraocular Muscle Fibres of Mammals. In P. Bach y Rita and C. C. Collins, editors, *The Control of Eye Movements*, pages 47–66, Smith-Kettlewell Institute of Visual Sciences, University of the Pacific Graduate School of Medical Sciences, San Francisco, California, 1971. Academic Press Inc.

[90] M. H. Pirenne. *Vision and the Eye*, chapter The Eye and the Formation of the Retinal Image, pages 1–24. Chapman and Hall LTD, 1967.

[91] J. Pola and H. J. Wyatt. Smooth Pursuit: Response Characteristics, Stimuli and Mechanisms. In R. H. S. Carpenter, editor, *Vision and Visual Dysfunction - Eye Movements*, volume 8, pages 138–156. Macmillan Press LTD, 1991.

[92] J. Porrill, P. A. Warren, and P. Dean. A Simple Control Law Generates Listing's Positions in a Detailed Model of the Extra-Ocular Muscle System. *Vision Research*, 40(27):3743–3758, November 2000.

[93] R. P. N Rao, G. J. Zelinsky, M. M. Hayhoe, and D. H. Ballard. Modelling Saccadic Targeting in Visual Search. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8 - Proceedings of the 1995 Conference*, pages 830–836. MIT Press, 1996.

[94] C. Rashbass. The Relationship between Saccadic and Smooth Tracking Eye Movements. *Journal of Physiology*, 159:326–338, June 1961.

[95] C. Rashbass. Second Thoughts on Smooth Pursuit. In P. Bach y Rita and C. C. Collins, editors, *The Control of Eye Movements*, pages 445–446, Smith-Kettlewell Institute of Visual Sciences, University of the Pacific Graduate School of Medical Sciences, San Francisco, California, 1971. Academic Press Inc.

[96] F. Reinhardt and H. Soeder. *dtv-Atlas zur Mathematik Tafeln und Texte*, volume 1. Deutscher Taschenbuch Verlag, 8th edition, June 1990.

[97] F. Reinhardt and H. Soeder. *dtv-Atlas zur Mathematik Tafeln und Texte*, volume 2. Deutscher Taschenbuch Verlag, 7th edition, July 1990.

[98] H. Ritter, T. Martinez, and K. Schulten. *Neural Computation and Self-Organizing Maps: An Introduction*, chapter The Oculomotor Control, pages 141–161. Addison-Wesley, 1992.

[99] D. A. Robinson. The Mechanics of Human Saccadic Eye Movement. *Physiology*, 174:245–265, April 1964.

[100] D. A. Robinson. The Mechanics of Human Smooth Pursuit Eye Movement. *Physiology*, 180:569–591, January 1965.

[101] D. A. Robinson. The Oculomotor Control System: A Review. In *Proceedings of the IEEE*, volume 56, pages 1032–1049, June 1968.

[102] D. A. Robinson. Models of Oculomotor Neural Organization. In P. Bach y Rita and C. C. Collins, editors, *The Control of Eye Movements*, pages 519–538, Smith-Kettlewell Institute of Visual Sciences, University of the Pacific Graduate School of Medical Sciences, San Francisco, California, 1971. Academic Press Inc.

[103] D. A. Robinson. Eye Movements Evoked by Collicular Stimulation in the Alert Monkey. *Vision Research*, 12:1795–1808, March 1972.

[104] D. A. Robinson. Oculomotor Control Signals. In G. Lennerstrand and P. Bach y Rita, editors, *Basic Mechanisms of Ocular Motility and their Clinical Implications*, pages 337–378. Pergamon Press, June 1974.

[105] RS Data Sheet: How to Select your RS Solenoid. Internet, March 1998. **http://www1.rswww.com/** 01.04.2002.

[106] G. Sandini. Log-Polar Mapping made Easy. Internet, 1995. **http://www.ai.mit.edu/people/giulio/logp.html** 01.04.2002.

[107] P. H. Schiller. A Model for the Generation of Visually Guided Saccadic Eye Movements. In D. Rose and V. G. Dobson, editors, *Models of the Visual Cortex*, pages 62–70. John Wiley & Sons LTD, 1985.

[108] H. Schmidt-Cornelius. Tangent Point Tracking for the Driving Task. In F. P. Retkowsky, editor, *The 11th White House Papers*, number CSRP 495, pages 69–78, School of Cognitive and Computing Sciences, University of Sussex, Brighton BN1 9QH, UK, October 1998.

[109] H. Schmidt-Cornelius. The Eye. In D. Pearce, editor, *The 12th White House Papers*, number CSRP 512, pages 56–57, School of Cognitive and Computing Sciences, University of Sussex, Brighton BN1 9QH, UK, November 1999.

[110] C. A. Scudder. A New Local Feedback Model of the Saccadic Burst Generator. *Journal of Neurophysiology*, 59(5):1455–1475, May 1988.

[111] C. Serafin. Preliminary Examination of Driver Eye Fixations on Rural Roads: Insight into Safe Driving. Technical Report UMTRI-93-29, University of Michigan Transport Research Institute, July 1993. Agreement Officer's Technical Representative (AOTR): Jerry Reagan, HSR-20.

[112] P. M. Sharkey, D. W. Murray, S. Vandevelde, I. D. Reid, and P. F. McLauchlan. A Modular Head/Eye Platform for Real-Time Reactive Vision. *Mechatronics*, 3(4):517–535, December 1993.

[113] H. J. Simonsz and I. D. Tonkelaar. 19th Century Mechanical Models of Eye Movements, Donders' Law, Listing's Law and Helmholtz' Direction Circles. *Documenta Ophthalmologica*, 74:95–112, February 1990.

[114] M. V. Srinivasan and S. Venkatesh. Introduction. In M. V. Srinivasan and S. Venkatesh, editors, *From Living Eyes to Seeing Machines*, pages 1–15. Oxford University Press, 1997.

[115] L. Stark. The Control System for Versional Eye Movements. In P. Bach y Rita and C. C. Collins, editors, *The Control of Eye Movements*, pages 363–428, Smith-Kettlewell Institute of Visual Sciences, University of the Pacific Graduate School of Medical Sciences, San Francisco, California, 1971. Academic Press Inc.

[116] O. Sutherland, H. Truong, S. Rougeaux, S. Abdallah, and A. Zelinsky. Advancing Active Vision Systems by Improved Design and Control. In *Proceedings of International Symposium on Experimental Robotics (ISER2000), Honolulu*, December 2000.

[117] M. Tistarelli and G. Sandini. On the Advantages of Polar and Log-Polar Mapping for Direct Estimation of Time-to-impact from Optic Flow. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(4):401–410, April 1993.

[118] C. W. Tyler. The Horopter and Binocular Fusion. In D. Regan, editor, *Vision and Visual Dysfunction - Binocular Vision*, volume 9, pages 19–37. Macmillan Press LTD, 1991.

[119] Author Unknown. The History of Electricity. Internet. **http://historia.et.tudelft.nl/wggesch/geschiedenis/electricity/** 01.04.2002.

[120] Author Unknown. The History of the Computer. Internet. **http://historia.et.tudelft.nl/wggesch/geschiedenis/computer/** 01.04.2002.

[121] Author Unknown. The History of the Computer 1971 - 1976. Internet. **http://historia.et.tudelft.nl/wggesch/geschiedenis/microprocessor/1971/** 01.04.2002.

[122] S. Vogel. I2C-Bus Kernel Support for Linux. Internet. **http://voxel.at/prj/i2c/** 01.04.2002.

[123] A. Watt. *Fundamentals Of Three-Dimensional Computer Graphics*. Addison-Wesley, 1989.

[124] B. Webb and J. Hallam. How to Attract Females: Further Robotic Experiments in Cricket Phonotaxis. Technical Report 800, Department of Artificial Intelligence, University of Edinburgh, 5 Forrest Hill, Edinburgh, EH1 2QL, Scotland, March 1996.

[125] D. Wilkie. Shortage of Chemical Fuel as a cause of Fatigue: Studies by Nuclear Magnetic Resonance and Bicycle Ergometry. In *Human Muscle Fatigue: Physiological Mechanisms*. Pitman Medical, 1981.

[126] J. D. Wirtschafter. Neurophysiology and Central Pathways in Oculomotor Control: Physiology and Anatomy of Saccadic and Pursuit Eye Movements. In C. W. Johnston and F. J. Pirozzolo, editors, *Neuropsychology of Eye Movements*, pages 5–29. Lawrence Erlbaum Associates Inc., 1988.

[127] P. Bach y Rita. Neurophysiology of Eye Movements. In P. Bach y Rita and C. C. Collins, editors, *The Control of Eye Movements*, pages 7–45, Smith-Kettlewell Institute of Visual Sciences, University of the Pacific Graduate School of Medical Sciences, San Francisco, California, 1971. Academic Press Inc.

[128] D. Young. Representing Images for Computer Vision, Cognitive Science Research Paper. Technical Report CSRP 096, School of Cognitive and Computing Sciences, University of Sussex, Brighton BN1 9QH, UK, December 1987.

[129] D. Young. Formal Computational Skills Course Notes. School of Cognitive and Computing Sciences, University of Sussex, Brighton BN1 9QH, UK, 1999.

[130] D. Young. Straight Lines and Circles in the Log-Polar Image. Technical Report CSRP 522, School of Cognitive and Computing Sciences, University of Sussex, Brighton BN1 9QH, UK, July 2000.

[131] L. R. Young. Pursuit Eye Tracking Movements. In P. Bach y Rita and C. C. Collins, editors, *The Control of Eye Movements*, pages 429–443, Smith-Kettlewell Institute of Visual Sciences, University of the Pacific Graduate School of Medical Sciences, San Francisco, California, 1971. Academic Press Inc.

[132] L. R. Young and L. Stark. A Discrete Model for Eye Tracking Movements. *IEEE Transactions on Military Electronics*, MIL-7:113–115, April 1963.

[133] L. R. Young and L. Stark. Variable Feedback Experiments Testing a Sampled Data Model for Eye Tracking Movements. *IEEE Transactions on Human Factors in Electronics*, HEF-4:38–51, September 1963.

[134] B. L. Zuber, J. L. Semmlow, and L. Stark. Frequency Characteristics of the Saccadic Eye Movement. *Biophysical Journal*, 8:1288–1298, 1968.

[135] N. Zuech and R. K. Miller. Vision for Industrial Robots. In *Machine Vision*, pages 89–106. Prentice Hall, 1987.