A Miniature Space-Variant Active Vision System: **Cortex-I**

Benjamin B. Bederson
June 1992

A dissertation in the Department of Computer Science submitted to the faculty of the Graduate School of Arts and Sciences in partial fulfillment of the requirements for the degree of Doctor of Philosophy at New York University

Approved: ______________________________
Eric Schwartz
Research Advisor

Abstract

We have developed a prototype miniaturized active vision system whose sensor architecture is based on a logarithmically structured space-variant pixel geometry. A space-variant image's resolution changes across the image. Typically, the central part of the image has a very high resolution, and the resolution falls off gradually in the periphery. Our system integrates a miniature CCD-based camera, pan-tilt actuator, controller, general purpose processors and display. Due to the ability of space-variant sensors to cover large work-spaces, yet provide high acuity with an extremely small number of pixels, space-variant active vision system architectures provide the potential for radical reductions in system size and cost. We have realized this by creating an entire system that takes up less than a third of a cubic foot. In this thesis, we describe a prototype space-variant active vision system (**Cortex-I**) which performs such tasks as tracking moving objects and license plate reading, and functions as a video telephone.

We report on the design and construction of the camera (which is $8 \times 8 \times 8$mm), its readout, and a fast mapping algorithm to convert the uniform image to a space-variant image. We introduce a new miniature pan-tilt actuator, the Spherical Pointing Motor (SPM), which is $4 \times 5 \times 6$cm. The basic idea behind the SPM is to orient a permanent magnet to the magnetic field induced by three orthogonal coils by applying the appropriate ratio of currents to the coils. Finally, we present results of integrating the system with several applications. Potential application domains for systems of this type include vision systems for mobile robots and robot manipulators, traffic monitoring systems, security and surveillance, telerobotics, and consumer video communications.

The long-range goal of this project is to demonstrate that major new applications of robotics will become feasible when small low-cost machine vision systems can be mass-produced. This notion of "commodity robotics" is expected to parallel the impact of the personal computer, in the sense of opening up new application niches for what has until now been expensive and therefore limited technology.

Acknowledgements

This work is the result of a collaborative effort. It was a pleasure to have had the opportunity to work with such fine minds as I found in Eric Schwartz (my thesis advisor), Richard Wallace (my colleague and constant teacher), and Ping-Wen Ong (my fellow student and supporter).

Eric has been doing research leading up to this project for at least ten years. He found the funding for it, and had many of the critical ideas that got it moving. The Spherical Pointing Motor was his original idea, and only through (literally) years of bouncing it off each other, did we get it right.

Richard is largely responsible for the digital signal processing in the system. If it weren't for him, we wouldn't have a video display. He had infinite patience in helping me work through problems. He was always available to assist me, and helped to keep the project well directed.

Ping-wen came through at the last minute to integrate his license plate tracking software to work with **Cortex-I**. In addition, he was a motivating force in my studying Chinese. While not directly critical to this project, studying Chinese helped me keep my sanity during various traumas – such as when the first prototype went flying through the air as I got a great electric shock – a few hours before we departed for Chicago where **Cortex-I** was first publicly demonstrated.

Everyone at the NYU Robotics Lab was always optimistic and supportive as **Cortex-I** slowly took shape. Bud Mishra, especially, had good ideas, and was kind enough to read this thesis twice.

And of course, I thank my family, friends, and cat (Tria) who have had to put up with me as I've been counting down the months to finish this, and start the next phase of my life.

Contents

List of Figures

Chapter 1

Introduction

Computer vision is a field that typically requires very fast and expensive machines. By using a radically different image architecture along with novel imaging and actuation technologies, we have created a system that we strongly believe will allow us not to solve vision problems better than before, but to solve them dramatically faster, smaller and cheaper.

The key innovation is the use of *space-variant images*[1] which are images whose *resolution* changes across the image. The central part of the image has a very high resolution and the resolution falls off gradually in the periphery. The main advantage of this type of image is the small amount of data it contains while maintaining a high *field width* and central resolution.

This thesis discusses image processing with space-variant images, and a miniature space-variant *active vision* system, **Cortex-I**, that integrates a camera, a two degree-of-freedom motor to *actuate* the camera, and a loosely coupled parallel computing architecture to process the images and control the motor and camera.

1.1 Motivation

Traditionally, very little computer vision research has focused on real-time systems. Typical systems spend several seconds or even minutes on solitary images. This approach has not produced a general real-time device. Part of the problem is the philosophy often followed of "we need to do basic research, so we'll write the necessary software and the hardware will eventually catch up."

This ignores the fundamental problem of bandwidth. Using a standard 512×512 image at video rates of 30 frames per second, 8 megapixels per second need to be processed. Even with specialized image-processing hardware (e.g., DataCube's MaxVideo system), the best general-purpose systems are still processing only the simplest scenes successfully in real-time. It is not likely that any technology in the near future will be able to supply the required processing power.

This problem is not unavoidable because typically, only a small part of the image contains the feature being examined at the moment. What is needed is a way to selectively focus *attention* on the interesting part of the image while maintaining some level of awareness of the rest of the image so interesting events can be recognized as such and then attended to.

Multiresolution images supply the opportunity to reduce bandwidth while allowing focusing of attention on one region and awareness of a wider field. Two relevant parameters of an image are the field width and the resolution. The field width specifies how large a portion of the scene is contained in the image. The resolution specifies how much information is contained in a fixed portion of the image. Neither of these parameters can be arbitrarily reduced without severely degrading the system performance. The human visual system solves the bandwidth problem without compromising either field width or *foveal*

[1] Emphasized words are defined in Appendix A

resolution. It does so with the use of the retina, which is a space-variant sensor. It has a resolution that is high in the center (called the *fovea*), low in the periphery and it changes smoothly between them.

Any system that uses a space-variant sensor must aim the sensor properly. Since space-variant sensors only have high resolution in the fovea, the current region of interest must be continuously tracked or *foveated*. Such a sensor must be mounted in a device that can aim the sensor with precision. A system with this capability is an example of an *active* or *attentive vision system*.

A *logmap* sensor is a useful type of space-variant sensor that is modelled on the human visual system (see Section 1.3). The space complexity aspect of logmap sensors is particularly attractive and has been analyzed in detail [42]. In this work, a *spatial quality measure*, Q_s, for sensors is defined to be the ratio of its work-space to maximal resolution. Logmap sensors can each achieve comparable Q_s to conventional uniform sensors that are one to four orders of magnitude larger. Image simulation of the human space-variant architecture suggests that this ratio may be as high as $10,000:1$ [49]. In current implementations of space-variant machine vision systems, logmap sensors with between 1000 and 2000 pixels have comparable Q_s to conventional sensors in the range of 256×256 to 512×512, a compression of between $60:1$ to $250:1$. Moreover, as shown in [42], Q_s grows exponentially with the number of pixels in the logmap sensor, thus providing a highly favorable route to upgrading sensor quality. The high compression ratios cited above for the human visual system derive from the fact that human acuity, which is roughly one-arc minute in the fovea, when extrapolated over a 120 degree visual field, yields a constant resolution sensor with roughly 0.2 gigapixels. The logmap sensor with comparable Q_s has roughly 10^4 to 10^5 pixels.

In this discussion, compression refers to the ratio of the number of pixels in a conventional uniform sensor to that of a space-variant sensor with the same Q_s. The issue of image compression, in the conventional (e.g. JPEG) usage is an independent issue not addressed in this thesis, but we point out that a form of *progressive video coding*, in which a video image is represented by a sequence of logmap images, is one application that has benefitted from the high compression ratio of the logmap transform. If each logmap image is centered on a different point in the video image and clipped so that we preserve the highest frequency information available at each point in the video picture, we can progressively construct a video image of increasing detail. Rojer [43] reported some experiments with progressive logmap video coding, and discussed the problem of selecting the best sequence of gaze points.

A space-variant sensor requires pointing the sensor at the region of interest. If the sensor is not pointed (or foveated) properly, the desired object will fall somewhere on the periphery of the sensor resulting in a lower resolution image. So it is important to develop efficient eye movement mechanisms.

A sensor movement system must find interesting objects, track them as they move, and account for movement of the device the sensor is mounted in. A short description of the way the human system solves the second two parts of this problem follows. It should be noted that the human visual system is able to track objects and foveate on them to within the accuracy of the retina's highest resolution. There are four principal types of eye movements as reviewed by Robinson [41].

1. **Saccades:** These are discrete movements and move the eye quickly ($300°$ - $400°$/second) from one fixation point to another. They also correct errors of the pursuit system. There is a large latency (150 ms) between fixations. There are typically less than four saccades per second.

2. **Pursuit Eye Movements:** The pursuit system tracks moving targets and keeps the current target foveated. It has a latency of 50ms.

3. **Vergence:** This controls the depth that the two eyes fixate on together. It is the slowest system and has a latency of over 200ms.

4. **Vestibular Systems:** The Vestibular Ocular Reflex and Optokinetic Reflex systems maintain the gaze of the eye, counteracting for head movements. I.e., if the head moves to the right, the eye moves to the left. This is one of the fastest types of eye movement and has a latency of only 14ms. These get sensory input from the semicircular canals that effectively constitute an angular inertial accelerometer with a frequency range from 0.017 to 17Hz.

In addition to these four systems, there is a fifth type of movement known as physiological nystagmus. This is a small higher frequency jitter at 30 to 80Hz. These movements are very important to the low-level visual process, but are not an issue in the larger movements of object tracking with which we are concerned here.

In our work, we focus on saccadic motions which are the simplest to implement and are used for both attention and tracking.

The mapping of the retina to the visual cortex is another very important characteristic of the human visual system. Pioneering work by Hubel and Wiesel [26,27] showed how some low-level feature detection performed on the retinal image was mapped to the V1 area in the visual cortex. Then Schwartz showed in [47,48] that the mapping from the retina to the visual cortex takes a very specialized form. Specifically, the retina can be looked at as a polar coordinate system where each point represents one photo-sensitive cell. It gets mapped to what is essentially a rectangular coordinate system in the visual cortex. This can be represented mathematically by a complex logarithm and is depicted graphically in Figure 1.1. This mapping is called the logmap.

The point in retinal space, P_1, can be denoted $re^{i\theta}$ and the point in cortical space, P_2, can be denoted by the complex number $x + iy$. If we take the log of P_1, we get:

$$\log(re^{i\theta}) = \log r + i\theta$$

Substituting

$$x = \log r \quad \text{and} \quad y = \theta,$$

we get

$$\log(re^{i\theta}) = x + iy$$

or,

$$\log(P_1) = P_2.$$

So the natural logarithm of a point in polar coordinates is transformed to a point in rectangular coordinates.

This mapping has some interesting geometric properties. Specifically, fovea-centered circles get mapped to vertical lines and radial lines get mapped to horizontal lines. This has the effect of transforming scale and rotation in fovea-centered retinal images to translation in cortical images. For many computer vision applications, it is easier to recognize translation of an object than scaling or rotation, so this provides another justification for a complex log sensor geometry. However, since objects get distorted as they move away from the fovea, it is practical to use a tracking system to keep the object of interest centered on the fovea. In the present work, we do not attempt to make use of these geometric features of the complex log mapping, but rather, we use this mapping for its inherent bandwidth reduction.

One problem in using an analytic space-variant mapping, such as the logarithm is the existence of a singularity at the origin of the coordinate system. Figure 1.1b shows this singularity. The left points of each triangle in the range all represent the same single point in the domain. Our solution to this problem is to introduce a small real constant, α, into the mapping and to use a map function of the form $\log(z + \alpha)$. The constant, α, is used in an analagous fashion in models of the human visual system: it is a measure of the size of the central linear representation of the human fovea [48]. There are still difficulties with the $\log(z + \alpha)$ mapping because there is a discontinuity in the range. This discontinuity, however, is much easier to handle than the singularity in the original mapping.

As can be seen in Figure 1.1d, the pixels on the edge are cut off in different places yielding 3, 4, and sometimes 5-sided pixels. This makes operations such as convolution difficult as it is not immediately clear how to handle the missing and extra adjacent pixels. We discuss a solution to this problem in Section 1.4

We have assumed that an interesting object has been located. But how do we find interesting objects throughout the visual field? This question is a very important one and is called *attention*. Yarbus [58] examined this problem in humans by tracking eye movements in individuals as they examined a scene.

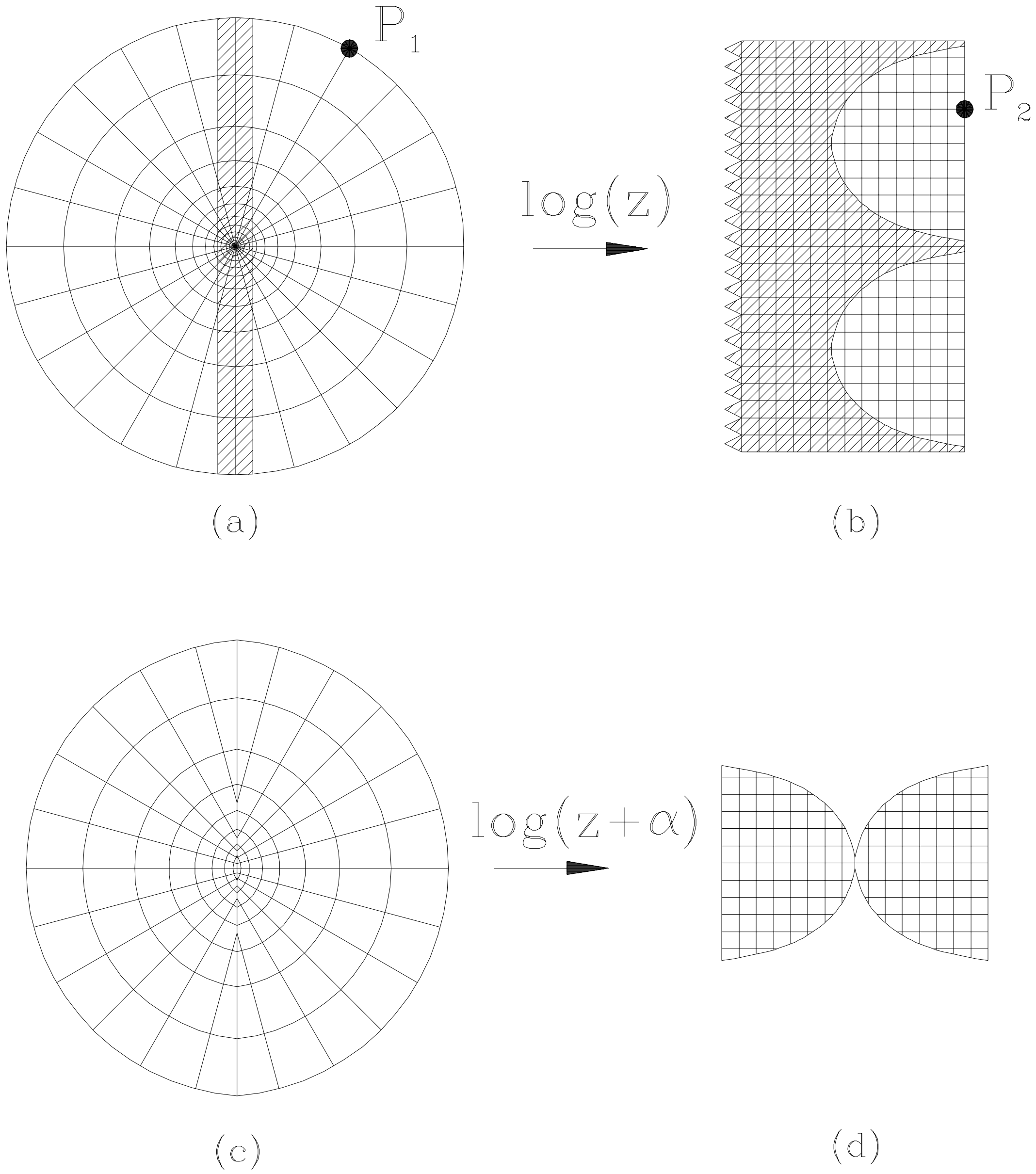


Figure 1.1: This shows the complex logmap, $log(z)$ in **a)** and **b)** and its relative, the mapping, $log(z+\alpha)$ in **c)** and **d)**. It can be seen that the $log(z+\alpha)$ map results from cutting the hatched region out of the $log(z)$ map. Note that rings centered around the fovea in the domain get mapped to vertical lines in the range and radial lines get mapped to horizontal lines. Here, P_1 gets mapped to P_2. The $log(z)$ map has a singularity in the center while the $log(z+\alpha)$ map has a discontinuity along the vertical meridian.

From this and other experiments, Yarbus showed that individuals tend to have consistent eye movement patterns that are different across individuals. He also showed that with supervision and a high-level goal, eye movement patterns (mainly saccades) could be controlled. For instance, in one experiment, Yarbus showed a picture to a subject and asked the subject to remember the position of the people and objects in the room, the response was a very unfocused scan covering the entire image. On the other hand, when the subject was requested to estimate how long the unexpected visitor had been away from the family, the response was to spend almost all the time examining the faces of the people. These experiments are important because they show how low-level control of the visual system is influenced by high-level processes.

An excellent review of selective fixation methods in both human and computational systems can be found in Abbott [1]. Two researchers not covered in this review are Rojer and Califano.

Rojer [43] addressed the issue of attention using space-variant images and showed several strategies for determining where to look next. He first compared several heuristic algorithms for attention. He quantified the results for attention by random fixation with and without inhibition, attention from a spatial derivative, and attention based on motion and regions. He then introduced a variation on the Hough transform that he calls the Collapsed Hough transform. It allows the feature to be of high dimension while using a "motor space" of only two dimensions, the minimum necessary to control fixation. He demonstrates results with this method to locate faces in a natural scene.

Califano et. al. [14] introduces a technique for foveation based on a two-level resolution system. They have implemented three modes of attention. One detects conflicts between object models, typically occurring when multiple objects differ only upon examination at a higher resolution. The second looks for areas that are not currently covered by any hypothesis. The last explores previously unexamined areas, much as some of Rojer's heuristic procedures do.

1.2 Background

Researchers have been examining the problems associated with space-variant sensors and active vision for many years, both in theoretical and practical ways.

On the theoretical side, Aloimonos et. al. looked at many of the shape-from-X problems [5], proving that an active search strategy is more efficient than a passive one, and that ill-posed and nonlinear problems become well-posed and linear. Tsotsos analyzed the relative complexity of active versus passive search [54]. He claims that an active search strategy may gain improvements in computation time based on worst-case time complexity functions he derived using hypothesize-and-test strategies.

Many groups have built active vision systems to experiment with some of these ideas. Krotkov [33] built what is often regarded as the first "robot head". Many other groups as detailed in Chapter 4 followed suit. The four groups described here have all built working systems that use some sort of multi-resolution technique, or in Dickmanns case, multiple foveation points.

At the Universities of Genoa and Pisa in Italy, Van der Spiegel et. al. and Sandini et. al. are doing collaborative work with Bajcsy at the University of Pennsylvania [6,32,45,46,55]. They have built a space-variant sensor and are working on geometrical and tracking algorithms for it.

Van der Spiegel et. al. has been developing a VLSI space-variant sensor based on the complex-log map, $\log(z)$. It is based on *Charge Coupled Device* (CCD) technology. They fabricated a working chip that produces images with 30 rings of 64 pixels each with a uniform fovea of 102 pixels. Each pixel gives a linear output in response to incident light intensity.

Because CCD technology will not allow direct connection between greatly different sized components, the design must be broken up into several stages, each normalizing its data before passing it on to the next. In addition, there is a limit to the size of the photo-sensitive area. These characteristics led to a design consisting of three sets of ten rings. Each new set of rings has twice as many pixel cells, each half as big as the previous set necessitating different clocks for each set of rings. The design also has a "blind spot" where the signals necessary for the center of the chip access it. The blind spot is actually one ray that is the width of 2.5 pixels.

Sandini et. al. simulated logmap images by sub-sampling *uniform images*. They have examined three problems: 1) recognition of arbitrarily scaled and rotated 2D shapes; 2) tracking of a moving target; and 3) evaluation of distance of an object using optical flow. All three problems take advantage of the space-variant structure of the image. These experiments only used the peripheral pixels and did not take into account the uniform fovea. They did, however, simulate the blind spot accurately.

The first problem Sandini examined is based on the standard complex logmap that converts rotation and scaling to translation for foveated scenes. Using simple binary objects, they implemented a template matching algorithm based on cross-correlation and achieved a recognition rate of 97% for 50 objects that were arbitrarily rotated and scaled. They claim that the errors are largely due to the blind spot.

The second problem, that of tracking a moving target, was examined with the intent of seeing how the low resolution of the image affected the system. The target used was a bright spot over a dark background moving in parabolic and circular paths. The target was identified by subtracting the background velocity (computed directly from the known camera motion) from the optic flow. This yielded a velocity estimate for the target that was located by computing the center of mass. The camera was then positioned based on the velocity of the target. When the error reached a threshold, a "saccade" was performed correcting the error.

For the third problem, they assumed the camera was moving with pure translation towards a foveated object with constant speed. They wanted to compute the distance of the object from the camera. The optical flow vectors for this situation are radial pointing outwards for a uniform image. For the logmap image, however, the vectors are horizontal lines since radial lines get mapped to horizontal lines in the $\log(z)$ map that they use. The vector's amplitude is constant across the image and is proportional to the distance the object has moved.

These preliminary experiments show how the geometrical structure of the logmap image affects some basic vision problems. The low resolution of the image does not seem to be a problem for this simple situation, although the blind spot does cause some difficulties.

Dickmanns and Graefe at the Universität der Bundeswehr München [20] have shown their real-time vision system to work in non-trivial difficult situations. Their approach is to maintain a world model of the situation, focus processing on selected regions, and use the temporal proximity of the images to incrementally process each new image. They find that the faster they run the system, the easier the processing becomes. As the images get closer together temporally, the difficulty of the correspondence problem decreases as the images change less and less between frames. They have demonstrated three working systems. One drives a car autonomously on an autobahn construction site. Another balances a stick in one-dimension on a moving cart. The third docks a vehicle on an airbed.

They define their task as looking "for the inherent structure and ...to find a computer architecture which is well matched to the task of visual motion control." [21, p. 224]. The most important concept is that of *temporal continuity* which is the basic idea that if two pictures of a natural scene are taken within a few milliseconds of each other, they will normally be very similar to each other. They have found that most features don't move more than a pixel or two between frames.

Another important concept is that the appropriate behavior of a system typically depends on only on a small number of features. Only the area of the image containing these features needs to be examined. This area is often less than 10% of the whole image. In a dynamic system, the location of these features in usually known with fairly good precision since they probably change very little between successive images. This approach influences the choice of hardware as each region of interest is best processed in different ways. A dynamic system is better off with a coarse grained parallelism with one processor handling each region of interest. Massively parallel SIMD machines, on the other hand, are not well suited to this approach since they would have to process all pixels the same way. They would waste 90% of their time processing the uninteresting regions. In addition, SIMD machines have no mechanism for processing different regions with different algorithms at the same time.

The final idea in the work of Dickmanns and Graefe is that of a world model. Their systems work in limited domains, and are not intended to work generally. The autonomous vehicle, for instance, works only on autobahns in good condition without other cars. Because of this, a world model can be accurately created that describes what is likely to be seen and how objects in the model are likely to act and react. The vehicle system knows about lane divider lines in the road, the edge of the road, side roads, road curvature, etc. It can predict how things are likely to change based on system state parameters (i.e., the speed of the vehicle). With each new image, the locations and shapes of the features can be accurately predicted and the world model can be updated.

The world model of Dickmanns and Graefe is a full 3D model while the images are 2D perspective projections of the world. The conversion from the 2D projection to 3D is non-unique. They solve this by first taking the 2D projection of the world model to estimate where the features are. They use knowledge based feature detection on the image (to be described shortly), and finally use a recursive least squares state estimation technique to bypass the non-unique inversion of the perspective projection and update the world model. This is a Kalman filtering [34] approach based on the Jacobian of the image features.

The world model technique runs into problems in at least three situations that are: 1) analyzing the scene at startup; 2) losing the location of features due to extreme circumstances; and 3) when new objects appear in the scene. The initialization is not difficult because that is not time critical. The other two situations, however, are very difficult problems that have not been substantially addressed in this work.

Their knowledge based feature detection is based on the world model. They predict the approximate location of the feature and use a controlled correlation technique to locate the feature precisely. This substantially reduces the cost compared to a standard correlation. They only check a small area surrounding the predicted location of the feature. They reduce the problem to one dimension by checking a path orthogonal to the expected orientation of the feature. The result is an extremely fast updating of feature position and shape.

The architecture of the system is based on up to fourteen 8086-based single board computers. Each processor examines a single region of interest. There is also one system processor that coordinates communication between the other processors. There is one wide-angle camera to examine the periphery and one camera with a telephoto lens to examine specific details. They both have pan and tilt control. The system has a video bus that can transmit up to four images simultaneously.

They have autonomously driven a small van at speeds of up to 60 mph on an autobahn construction site with no other vehicles. The van has run 20 km without human intervention in clear and moderately rainy weather. It is able to enter the road from acceleration strips, switch lanes, and go on and off exit ramps.

This approach clearly has merit and has been demonstrated to work in a variety of relatively uncluttered environments where a fairly simple and accurate model of the world exists. Its ability to generalize to complex systems was not demonstrated.

A large research group directed by Brown and Ballard at the University of Rochester [13] are exploring some of the basic concepts of active vision systems centering on eye and body movements. Their stated goals are to study real-time vision in cognition and movement.

Their system, called "The Rochester Robot", consists of two cameras mounted on a Puma robot. Each camera has separate pan and common tilt control. The Puma can move the "head" in a two meter radius with six degrees of freedom.

Their approach in fighting the bandwidth problem is to use special-purpose hardware. They are using a DataCube MaxVideo image processing system for low-level image processing and a 128 processor BBN Butterfly computer. They incorporated a second camera to have two different resolutions. One camera has a high fieldwidth and low resolution while the other has low fieldwidth and high resolution. They also talked about building a *pyramid* architecture with five levels of 64×64 images where each level subsamples the input image at a different rate.

One of the problems they have examined is that of automatically maintaining vergence on an object as the body moves. They are also studying saccade and pursuit mechanisms. Finally, they are exploring the difference between self and world motion. They note that fixating systems have effective space-variant resolution due to blurring in the periphery. However, this is based on the finite integration times of the photo-sensitive elements of the sensor. It is not clear if egomotion is rapid enough to cause such blurring.

They create a depth map for a foveated object via motion parallax by moving the head sideways while fixating on a point. Objects in front of the point appear to move in the opposite direction of the head motion while objects behind the point move in the same direction as the head. The velocity of the object motion is proportional to the distance from the fixation point. By computing the disparities of the images taken from different positions (that they don't discuss), a depth map relative to the fixated point can be computed.

Another technique they discuss uses self-motion for dynamic segmentation. While moving and fixating on a point, the object that is fixated on becomes isolated because everything around it moves and blurs. They process this by computing a weighted average of input images over time. There are some potential problems with this technique as it is dependent proper fixation. They did not demonstrate any results with this technique.

They have also been investigating pursuit of moving objects. Their technique is based on correlating the image with a filter, which they can do very fast with the DataCube. However, their DataCube system can only perform up to 8×8 correlation at video rates. The result is that only small features that move slowly can be tracked.

Olson and Potter [36] have explored the problem of real-time vergence control: maintaining the gaze of two cameras on a single point in world coordinates. The Rochester Robot has one dominant eye that is under the control of the saccade and pursuit mechanisms. The dominant eye foveates an object, and the other eye is then aimed at the foveated object. The basic control loop grabs the left and right images, estimates the disparity, and re-aims the second eye. The difficult problem here is how to compute the disparity in real-time. They use a mechanism based on the Cepstral filter described by Yeshurun and Schwartz [59].

The Cepstral filter is computed by taking the power spectrum of the Fourier Transform of the log of the power spectrum of the Fourier Transform of the input image. The input image is a window from the left and right images spliced together. The results are peak values at $(\pm(w + d_h), \pm d_v)$ where d_h and d_v are the estimated horizontal and vertical disparities and w is the width of one image window. By analyzing the algorithm and implementing it very carefully on a Sun 3/260, they were able to speed it up from 2.6 seconds to under 0.1 seconds, which is fast enough for their system.

The Cepstral filter can be described in a couple of ways. Yeshurun and Schwartz describe it as extracting a periodic term in the log power spectrum of the original and shifted image. This periodic term corresponds to the disparity. Olson and Potter, on the other hand, prefer to think about it as a variation of the autocorrelation function where the log favors the signal with the broadest spectrum. These are impulses, so the result is an autocorrelation-like function that enhances sharp changes and is thus easier to interpret.

They have demonstrated results, but state that they still need to address the issue of the appropriate window size for the filter. They also want to work on integrating vergence with the saccadic and pursuit movements of the dominant eye. Although the individual parts of the Rochester Robot all work, they have not yet been fully integrated and have been tested only on simple geometric objects.

At the David Sarnoff Research Center, Burt built a working discrete approximation to a logmap space-variant sensor with a Laplacian pyramid approach. He [12] describes the sensor along with algorithms for its use in foveating, tracking, and interpretating objects.

A Gaussian pyramid is a set of images, each one half the resolution of the one before. A Laplacian pyramid is created by taking the difference between each level of a Gaussian pyramid. A full Laplacian pyramid contains all of the information in the original image. Burt's pyramidal approach is to take a set of Laplacian pyramid images and extract a window of the same size from each one. The result, called a *truncated pyramid*, is a set of images that take a finer and finer look at a smaller and smaller portion of the image. This approach is depicted in Figure 1.2. He constructed special-purpose hardware that computes this type of truncated pyramid. Crowley [17,18] also investigated pyramids for computer vision. He showed how multiple resolution pyramids can be used to match stereo scan lines.

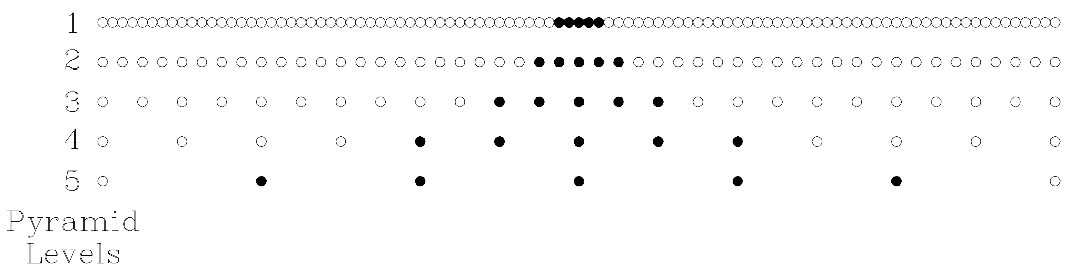


Figure 1.2: This shows the truncated pyramid structure used by Peter Burt. Each successive level represents a lower resolution image. The filled in circles represent pixels present in the pyramid. Notice that there are a constant number of pixels in the pyramid at each level where they cover a larger area at each level. This figure adapted from [12, p. 980].

The use of this architecture is based on coarse to fine analysis of an image. First, Burt picks a window size that can be processed by the CPU quickly. A window size of 16×16 is reasonable. His algorithm examines the image at the lowest level as this represents a low resolution representation of the entire image. If there is any evidence of a region of interest, then it examines the next level where a smaller area at higher resolution is represented. The algorithm climbs the pyramid in this manner with each new level directing the processing to a more specific area until the highest level is reached and the feature is fully processed. Assuming the original image is $512 \times 512 = 252,144$, and examining one window per frame, a feature will be fixated with the highest resolution after traversing all six levels. This involves processing $6 \times 16 \times 16 = 1536$ pixels, an effective data reduction of $252,144/1,536 \approx 164$.

This technique effectively replaces mechanical eye movements with electronic ones. Instead of physically moving a sensor, a region is examined within a fixed sensor. This technique obviously is much faster than a mechanical system, but is limited to examining scenes that fit within the limited field of a single image.

This architecture is able to track moving targets from a moving platform, such as when viewing a moving car from an airplane. A traditional approach is to analyze the motion between each pair of successive frames. This is computationally expensive and error prone if the relative motion of the object is small compared to the camera. Instead, if the relative motion of the background can be computed, then it can be subtracted out between successive frames and the slowly moving object will stand out. This is similar to how the human system operates. If a person is looking out a car window, s/he will often track a fixed object at which point relatively slowly moving objects will stand out.

The algorithm used to estimate the background motion is to first pick a level in the pyramid, k. The algorithm computes the displacement between frames 1 and 2 at level k. This is a relatively simple step because of the reduced resolution at this level. This computed displacement is a first-order estimate that can then be refined. Frame 2 is now shifted by this displacement but because of the low resolution, will not match exactly with frame 3. The algorithm computes the displacement at level $k-1$ and this process continues it reaches a good estimate. With nearly constant velocity, only two or three iterations have been needed to lock onto the background motion at full image resolution. As pointed out by Ballard and Brown, the foveated objects can then be recognized by dynamic segmentation.

Burt is using a pyramidal architecture that approximates a discrete version of a complex log mapping, $\log(z)$. This architecture allows him to choose the resolution appropriate for the current task, but with that comes the extra burden of always having to choose the correct level to use. This system is not mounted on a movable platform and no real-time results of the overall system were reported.

A fairly common solution to the problem of not having enough resolution and field-of-view at the same time is to use two cameras, one with a large field-of-view, and one with a small one. Burt [12, p. 979] points out that this approach will give trouble when attempting to foveate an object. Suppose that an interesting feature is seen in the periphery and the system needs to foveate on it. Since only low resolution data is available, a rough estimate can be made to re-aim the camera, but it is possible that the feature will still be outside the fovea. There will be no new information and thus it something of a hit or miss situation. This is opposed to a smoothly space-varying sensor (such as one with a logmap architecture) where after the first movement, even if the feature is not fully foveated, it will fall on a higher resolution portion of the sensor that will enable the system to make a second movement based on more accurate information. The system will quickly converge yielding a well-foveated feature.

1.3 Space-variant images

A space-variant image sensor is characterized by an irregular pixel geometry. We have experimented with several sensor designs, the common features of which are large scale changes between the smallest and largest pixels, wide field of view, and a pixel population far smaller than that of a conventional uniform image sensor. One such space-variant sensor is the complex log mapping, $\log(z + \alpha)$, in which the pixel pattern approximates the sensor geometry of the human eye [42,43].

We often want to display a logmap image on a TV screen, or to compute one from a *TV image*. Thus we can formally define the logmap as a mapping from a TV image, $I(i, j)$, where $i \in \{0, \ldots, \mathsf{NROWS}-1\}$ and $j \in \{0, \ldots, \mathsf{NCOLS} - 1\}$. Let $L(u, v)$ be the logmap, with $u \in \{0, ..., \mathsf{NSPOKES} - 1\}$ and $v \in \{0, ..., \mathsf{NRINGS} - 1\}$.

The forward mapping from TV image space to logmap space is specified by the *spoke and ring lookup tables*, $S(i, j)$ and $R(i, j)$ (Figure 1.3), where again $i \in \{0, \ldots, \mathsf{NROWS}-1\}$ and $j \in \{0, \ldots, \mathsf{NCOLS}-1\}$. Let $a(u, v)$ be the area (in TV pixels) of a logmap pixel (u, v).

$$a(u, v) = \sum_{i,j} 1 \mid S(i, j) = u \text{ and } R(i, j) = v. \tag{1}$$

The logmap (or *forward map*) image (Figure 1.4c) is defined by

$$L(u, v) = \frac{1}{a(u, v)} \sum_{i,j} I(i, j) \mid S(i, j) = u \text{ and } R(i, j) = v. \tag{2}$$

The *inverse map*, illustrated in Figure 1.4b, is

$$L^{-1}(i, j) = L(S(i, j), R(i, j)). \tag{3}$$

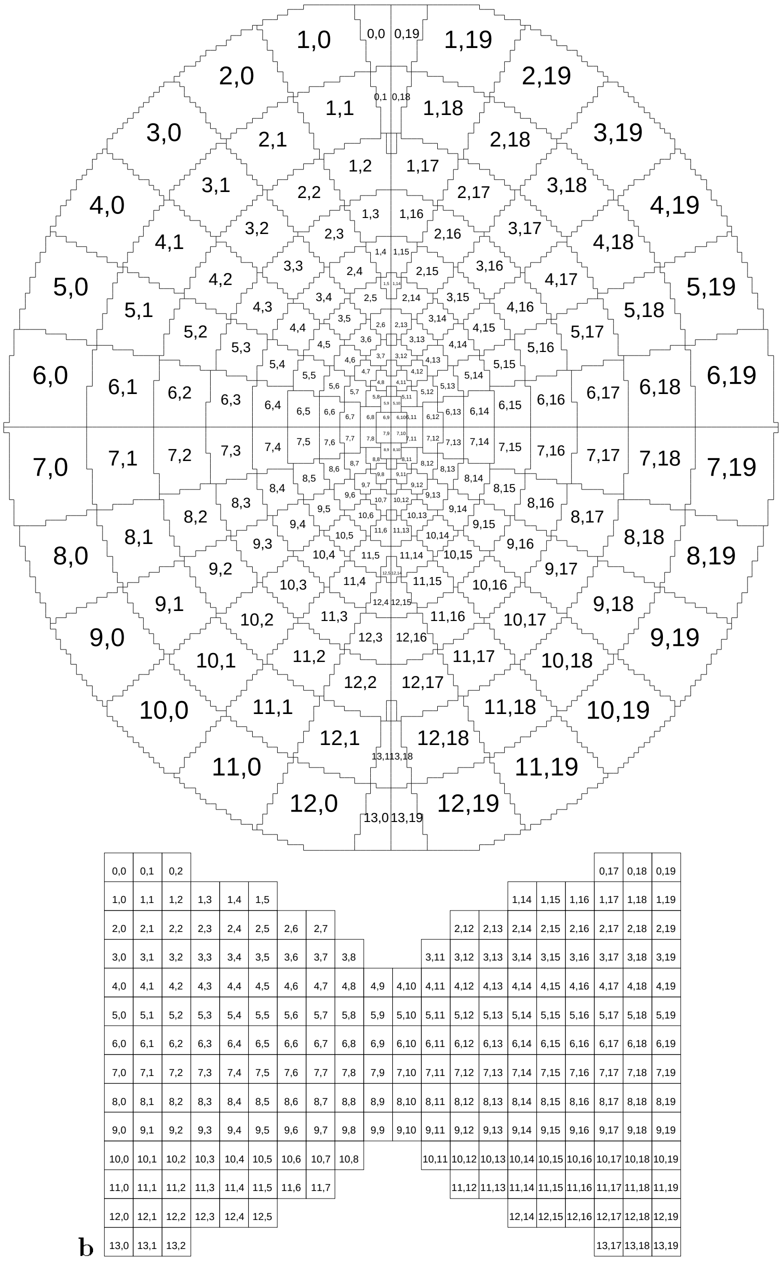
The relationship between a space-variant image and the lookup tables $S(i, j)$ and $R(i, j)$ is illustrated by comparing Figure 1.3 with Figure 1.4. The values in the lookup tables depict the row and column addresses of pixels in the space-variant image array. We observe that if n is the number pixels for which $a(u, v) > 0$, then $n \leq \mathsf{NSPOKES} \times \mathsf{NRINGS}$, and we define $\mathsf{NPIXELS} = n$.

We present a fast algorithm to compute the logmap from a TV image using $S(i, j)$ and $R(i, j)$ in Section 2.4.

1.4 Space-variant image processing

We are able to perform standard image processing algorithms on logmap images with the use of a *connectivity graph*. We define the connectivity graph, $G = (V, E)$, to be the graph whose vertices, V, stand for sensor pixels and the edges, E, represent the adjacency relations between pixels. Associated with a vertex p is a pixel address (u, v). Thus we write $(u(p), v(p))$ for a pixel coordinate identified by its graph vertex, or $p = \phi(u, v)$ for a vertex identified by its pixel coordinate. Then, $V = \{p_0, \ldots, p_{\mathsf{NPIXELS}-1}\}$.

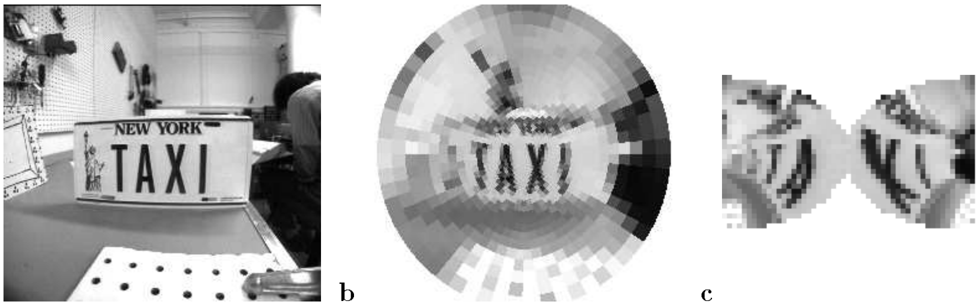
The definitions of $S(i, j)$ and $R(i, j)$ do not constrain the mapping. Let E_F be the set of edges representing connections between adjacent pixels in the forward map. The edge $(p, q) \in E_F$ provided



a

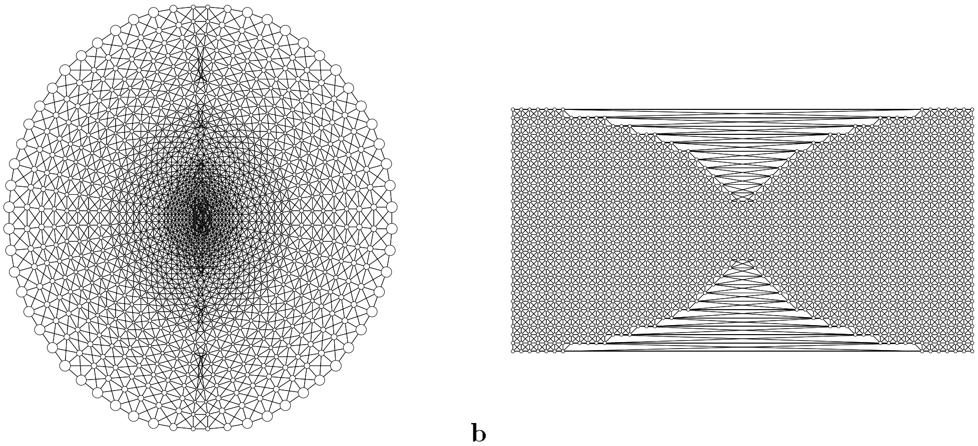
b

Figure 1.3: **a)** The lookup tables $S(i,j)$ and $R(i,j)$ combined into one figure. **b)** The forward map space. The correspondence between inverse and forward map pixels is shown.



a b c

Figure 1.4: A sample of **a)** a TV image, $I(i,j)$, **b)** The inverse logmap image $L^{-1}(i,j)$, and **c)** The forward logmap image $L(u,v)$.



a b

Figure 1.5: The connectivity graph for a $\log(z+\alpha)$ sensor geometry. Circles stand for graph vertices and lines stand for graph edges. The diameter of the circles in the inverse map is proportional to the area of the pixel. The graph explicitly represents the neighbor relations across the vertical meridian. **a)** The inverse map; and **b)** The forward map.

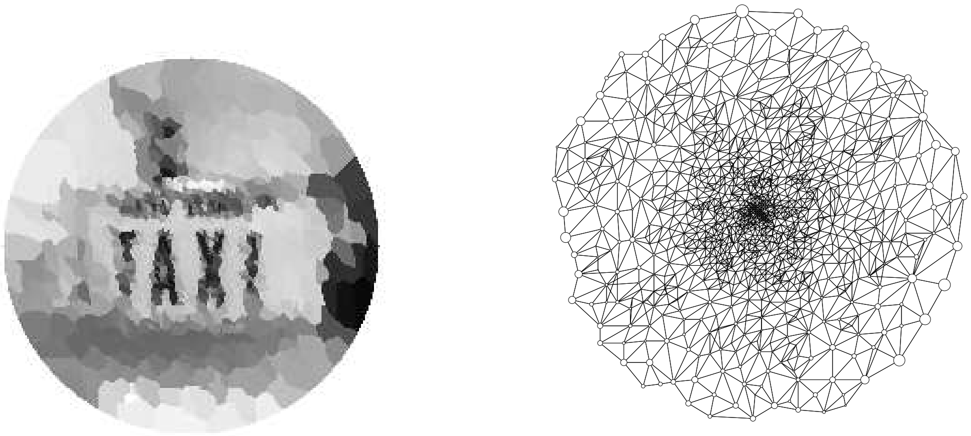


Figure 1.6: **a)** The image from a sensor having a random pixel geometry. **b)** The connectivity graph for this sensor.

$(u(p), v(p))$ is a neighbor of $(u(q), v(q))$. For the $\log(z+\alpha)$ mapping (Figure 1.1c) E_F does not contain the connections across the vertical meridian. To obtain all the connections in the sensor, we place the inverse image adjacency relations in the edge set E_I:

$$(p,q) \in E_I \quad \begin{array}{l} \text{provided } \exists\ i,j,k,l \text{ such that} \\ S(i,j) = u(p), R(i,j) = v(p) \text{ and} \\ S(k,l) = u(q), R(k,l) = v(q) \text{ and} \\ (i,j) \text{ is a neighbor of } (k,l) \end{array} \tag{4}$$

The connectivity graph edge set, E, may be $E = E_I$ or $E = E_I \cup E_F$, depending on the map. Resulting from the discretization of the complex log function $\log(z+\alpha)$ in the tables $S(i,j)$ and $R(i,j)$, some of the 8-adjacent neighbors in the forward map correspond to nonadjacent pixels in the inverse map. In this case, we chose $E = E_I \cup E_F$.

The algorithm to compute the connectivity graph first allocates one vertex, p, for each pixel (u, v). The graph edges in E_F are just the 8- or 4- edges from the forward map image. The graph edges in E_I are computed by scanning the lookup tables $S(i,j)$ and $R(i,j)$ with a 2×2 operator. The following algorithm, **compute_inverse_map_edges**, finds the edges E_I for the 4-neighbors.

Algorithm: **compute_inverse_map_edges**
Function: Computes the edges from the inverse map for the connectivity graph.
Input: Spoke and ring look-up-tables, $S(i,j)$ and $R(i,j)$
Output: Inverse map edges, E_I

$E_I = \{\}$
For each TV pixel (i,j)
  $u = S(i,j)$
  $v = R(i,j)$
  For each $(k,l) \in \{(i+1,j),(i,j+1),(i-1,j),(i,j-1)\}$
    $w = S(k,l)$
    $z = R(k,l)$
    If $(u \neq w$ **or** $v \neq z)$ **and** $e = (\phi(u,v),\phi(w,z)) \notin E_I$
    Then $E_I = E_I \cup \{e\}$.
  End For
End For

The **compute_inverse_map_edges** algorithm can be changed in a trivial way to pick up 8-neighbors. Another detail is that some points (i,j) may be outside the domain of the map, and $S(i,j)$ and $R(i,j)$ are undefined at those points. Also, this simplified algorithm does not take into account the fact that not all points (k,l) lie inside the bounds of the TV image.

Each graph vertex, p, in the connectivity graph is represented by a constant size data structure in memory. The graph edges are represented by one field containing a list of pointers. The structure for p may also contain fields such as the pixel array coordinates $(u(p),v(p))$, the pixel area $a(p)=a(u(p),v(p))$, etc. We define the pixel centroid $\boldsymbol{\mu}(p)$ to represent the centroid of the TV pixels that map to p.

$$\boldsymbol{\mu}(p) = \frac{1}{a(p)}\sum_{i,j}(i,j)^T \mid S(i,j) = u(p) \text{ and } R(i,j)=v(p). \tag{5}$$

We define the set of neighbors of p to be denoted by

$$\mathcal{N}(p) = \{q \mid (p,q)\in E\}. \tag{6}$$

The centroid and neighbors of a pixel are useful in writing image processing operations on space-variant images. For example, we can define a simple edge detector as

$$e(p) = \frac{1}{|\mathcal{N}(p)|}\sum_{q\in\mathcal{N}(p)}(L(p)-L(q))^2. \tag{7}$$

Note that the use of the neighbor-list allows us to define image processing operations on pixels having different numbers of neighbors. This eliminates the need for special cases for pixels at the boundary of the image.

We use the pixel centroid, $\boldsymbol{\mu}(p)$, to take into account the relative differences in pixel size. We illustrate this with a plane-fitting edge operator. Given $m = |\mathcal{N}(p)|+1$ points

$$\{r_0,\ldots,r_{m-1}\} = \{p\}\cup\mathcal{N}(p) \tag{8}$$

and their gray values, $L(r)$, we fit a plane $L=\mathbf{a}\cdot(i,j,1)^T$ to minimize some error measurement. Applying the least squares error criteria for linear equations, we can find a plane by solving for

$$\mathbf{A}=\mathbf{aB} \quad \text{where } \mathbf{A}=(L(r_0),\ldots,L(r_m)),\ B=\begin{pmatrix}\boldsymbol{\mu}(r_0) & \ldots & \boldsymbol{\mu}(r_m)\\ 1 & \ldots & 1\end{pmatrix} \tag{9}$$

The solution of $\mathbf{a}$ for $|\mathcal{N}(p)| \geq 2$ is

$$\mathbf{a} = \mathbf{A}\mathbf{B}^{\mathbf{T}}(\mathbf{B}\mathbf{B}^{\mathbf{T}})^{-\mathbf{1}} \tag{10}$$

If not all points are colinear, then $(\mathbf{B}\mathbf{B}^{\mathbf{T}})^{-\mathbf{1}}$ always exists. Since $\mathbf{M} = \mathbf{B}^{\mathbf{T}}(\mathbf{B}\mathbf{B}^{\mathbf{T}})^{-\mathbf{1}}$ is constant for each pixel, p, we precompute and store this matrix $\mathbf{M}(p)$ for each graph vertex p. At run time, we only need to compute $\mathbf{AM}(p)$ for every graph node p.

A simple relaxation procedure, the soap bubble algorithm, is also defined naturally on the connectivity graph. If p is a seed point, then $L_{t+1}(p) = L(p)$, otherwise,

$$L_{t+1}(p) = \frac{1}{1 + |\mathcal{N}(p)|}\left(L(p) + \sum_{q\in\mathcal{N}(p)} L_t(q)\right). \tag{11}$$

The definitions given so far have expressed functions of a pixel and only its immediate neighbors. Sometimes a local operator may map from a larger neighborhood, however. One operation helpful in defining such operations is the $\mathcal{N}_2$ function.

$$\mathcal{N}_2(p) = \{r \mid r \in \mathcal{N}(q) \text{ and } q \in \mathcal{N}(p) \text{ and } r \notin \mathcal{N}(p)\}. \tag{12}$$

We can define a space-variant sensor geometry that is not a complex logarithm, but is the result of a stochastic process. We define this map by selecting seed points randomly from a density function defined so that more points will be in the center of the image. We then create the map by growing each seed point until its neighbor is met using a simple relaxation procedure. The map is called the *random map* and is illustrated in Figure 1.6. To obtain the connectivity graph for this mapping, we use only $E = E_I$.

One application for the random map would be to use it with large area CCD image sensors with defective pixels. Large CCD sensors (1K×1K or 2K×2K pixels) are extremely expensive because of their low production yield. These sensors are much cheaper when they have defective pixels. We could create a random map with the associated connectivity graph for each of these sensors, skipping the bad pixels. This would allow for a very high resolution camera that would not be prohibitively expensive. We also could modify the $\log(z + \alpha)$ mapping to skip bad pixels in such a sensor.

A variety of space-variant sensor geometries have been discussed. Some have been simulated and some have been fabricated [15,42,43,45,46,55]. We can define a connectivity graph for any of these mappings, and then use our image processing library directly on these sensors. The connectivity graph provides a generic approach to image processing on an unconstrained sensor geometry [57].

1.5 System description

The **Cortex-I** (Figure 1.7) integrates a custom miniature logmap camera, *pan-tilt actuator*, controller, general purpose processors, and display. The entire system takes up a volume less than a third of a cubic foot.

The camera (Chapter 2) consists of a miniature commercially available CCD image sensor and a custom lens assembly mounted in the actuator. The camera image is mapped to a logmap image with a fast algorithm described in Section 2.4. Its maximum resolution is 0.175 degrees per pixel and its horizontal field of view measures 33^o. The sensor has a fixed focus 4mm lens with a manually changeable aperture (3 sizes). Imaged objects more than 40 mm away from the lens are in focus. The system outputs up to 30 frames per second and measures 256 gray levels per pixel. The camera head (CCD and lens assembly) measures only $8 \times 8 \times 10$ mm. A driver board controls the camera, providing timing signals to the sensor and converting analog sensor data to 8-bit digital data.

A preliminary actuator called the *Platform Pantilt* (Chapter 3) moves a platform by raising and lowering two shafts with linear stepper motors that are attached to the platform with universal joints. This actuator was built as a fall-back position before the Spherical Pointing Motor was operational. The Platform Pantilt measures $7.5 \times 7.5 \times 12$ cm and can move at rotational velocities up to 100^o/sec.

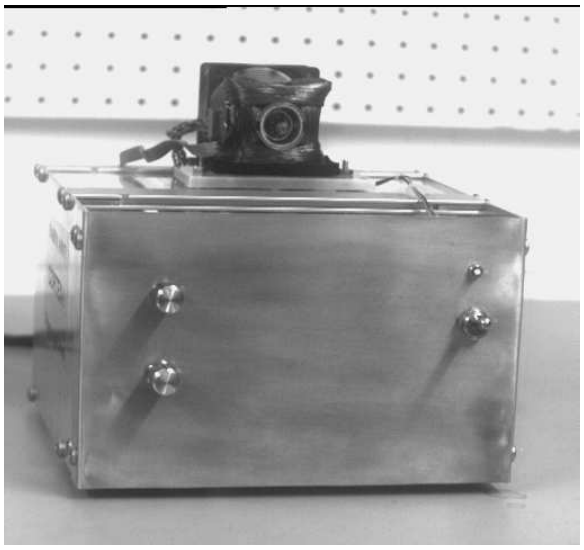


Figure 1.7: **Cortex-I**. Its dimensions are $14 \times 22 \times 22$ cm.

The *Spherical Pointing Motor* (SPM) replaces the Platform Pantilt because of its superior accuracy, speed, and size. The SPM (Chapter 4) is a novel pan-tilt actuator using three orthogonal motor windings to achieve open-loop pan-tilt actuation of the camera sensor in a small, low-power package. The SPM can orient the sensor through approximately 60^o pan and tilt, at speeds of several hundred degrees per second. It measures $4 \times 5 \times 6$ cm and its mass is 170 grams.

The controller consists of a camera driver, a 2 MIPS programmable microcontroller (Motorola M-C68332), a video display driver, a 12 MIPS digital signal processor (Analog Devices AD2101) DSP that maps the uniform camera image to a space-variant image, and an optional DSP board that acts as a video telephone. The actuator and camera are mounted to the chassis ($14 \times 22 \times 22$ cm) and connected by twisted-pair cables. The prototype is powered from a standard 110 Volt AC line, but uses less than 25 watts and could be battery powered.

The microcontroller controls the SPM and runs the application software. The MC68332 may be connected to a host processor for applications development, to upload sensor data, or to receive pan-tilt commands. The MC68332 is a 32-bit 16 MHz microcontroller integrating peripheral controls, such as programmable digital control lines and timing signals, directly on chip. We connected 192 Kbytes RAM and 128 Kbytes EPROM to the microcontroller. The microcontroller inputs the logmap data via a high speed serial interace from the DSP.

We have implemented image processing demonstrations in the microcontroller ROM. A simple motion tracking program, that turns the camera to center the observed motion field, runs at 16 frames per second (fps). The ROM also contains a test pattern, image binarization (22 fps) and a motor motion demo. We also use the connectivity graph [57], to implement image smoothing (3 fps), edge detection (2 fps) and connected components (1 fps), illustrating both the generality of the programming model and the limitations of the microcontroller in the current prototype.

The Analog Devices 2101 digital signal processor (DSP) reads the uniform CCD image and maps it to a space-variant image. It outputs the logmap over a high-speed serial connection. The DSP board combines an AD 2101 12 MHz DSP having two 2 MHz serial ports, 8 Kbytes internal RAM, 80 Kbytes external RAM, and 64 Kbytes boot EPROM. A second identical DSP board functions as a video display driver, generating an RS-170 video output suitable for display on a standard TV monitor. An optional third DSP board is used as a video transmitter that can send 4 frames per second of logmap images over a standard voice-grade telephone line.

Several applications using **Cortex-I** are described in Chapter 5.

Chapter 2

Camera

2.1 Introduction

The first problem to solve in building **Cortex-I** is how to acquire logmap images. We have pursued two routes to solve this problem. The first is to develop a custom VLSI sensor chip with a logmap geometry that will return a logmap image directly. The second is to use a commercially available sensor chip that returns conventional rectangular images and map them into logmap images.

The main advantages of a custom logmap sensor chip are the high frame rate and small resultant system size. This is because the geometry of the chip performs the mapping intrinsically. The sensor layouts that we have designed consist of several thousand pixels on chip, and this is a very small amount of pixels to digitize. It is likely that custom sensor chip could support rate of thousands of frames per second, contingent on the ambient light intensity. However, at the present time, we do not have a fully working space-variant chip.

The main advantages of an off-the-shelf conventional (uniform) sensor chip are low price, high image quality, and immediate availability. Because uniform sensors have been refined for many years and are produced in great quantities, they have extremely high quality which is difficult to match with custom sensor chips. The image from a uniform sensor, however, must be mapped to a logmap image according to the definitions in Section 1.3, and this is computationally expensive since the final logmap image size of thousands of pixels must be supplied through a read-out bottleneck of hundreds of thousands of pixels on the chip. However, with careful design of the readout and logmap emulation algorithms, we have achieved frame rates of up to 30 frames per second, without resorting to excessively complex readout electronics.

We are currently developing a custom logmap sensor chip in collaboration with Synaptics, Inc. The first prototype based on *CMOS* technology has about 500 pixels and does not work to our satisfaction. Figure 2.1 shows some images from this chip. We are currently working on both improving the quality of the image, and increasing the number of pixels.

We also developed a camera based on a uniform sensor chip designed to be mounted on the Spherical Pointing Motor. It uses a low-resolution area CCD sensor, and returns a digital image directly to the computer in our vision system which is then mapped to a logmap image. We made two lenses. One has a fixed field of view (Figure 2.2) and the other has an actuated two position zoom lens (Figure 2.3).

Most other computer vision researchers have used off-the-shelf video cameras for their vision systems. This is certainly a reasonable route to take as it avoids the cost and time involved in developing a custom camera. One group [32,46,55], however, has developed a custom CCD logmap sensor. Like us, they have found it a very time consuming and difficult path. Their current sensor has approximately $2,000$ pixels and returns reasonable images.

So why build our own camera? Simply stated, the reason is size. This includes both the size of the

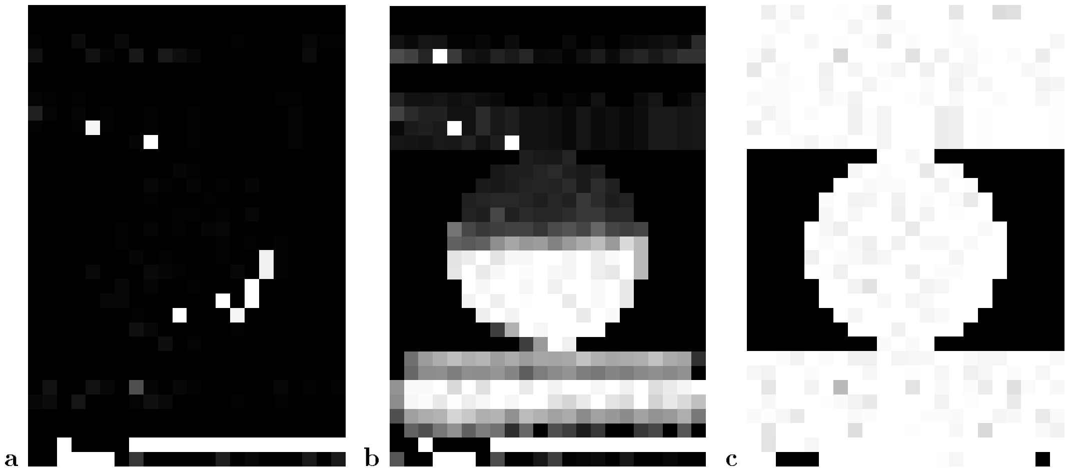


Figure 2.1: Images from our custom logmap sensor chip. They are the logmap images from the chip, and have not been mapped back to scene coordinates. They are scenes of **a)** an all black scene, **b)** a horizontal edge with the top half black and the bottom half white, and **c)** an all white scene. The circle in the center of the images is an artifact of the chip geometry.

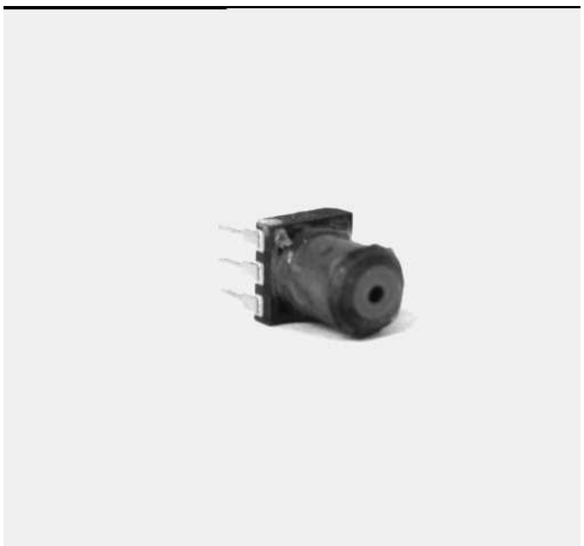
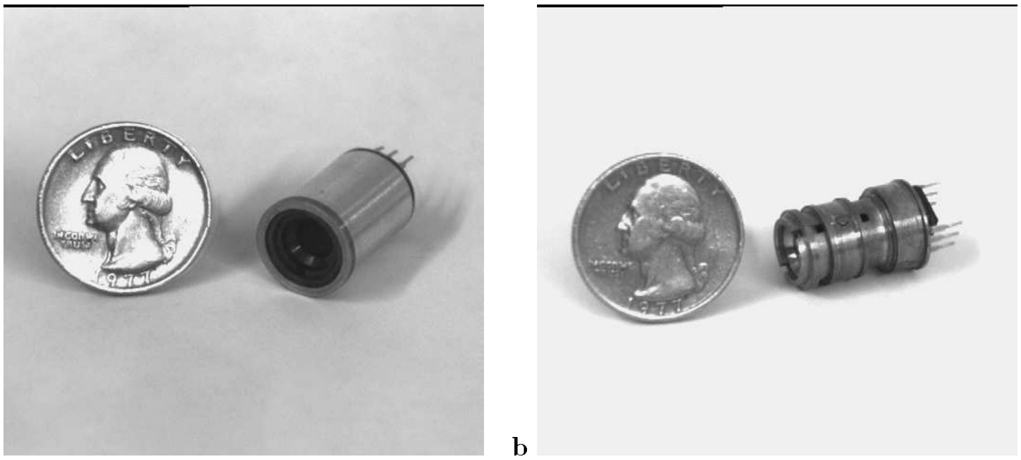


Figure 2.2: The fixed field of view camera. It consists of a four element lens assembly attached directly to a CCD sensor.



a **b**

Figure 2.3: The zoom camera. It consists of a six element zoom lens assembly including a coil to actuate the zoom lens. It is attached directly to the CCD sensor. It is shown **a)** with and **b)** without the light baffle.

camera head, and the necessary associated readout electronics. Even the smallest commercial cameras available today have camera heads that are several centimeters long and weigh hundreds of grams. To satisfy the goals of this project, we need a camera head that is about one centimeter long and that weighs a couple of tens of grams. Because we are using space-variant images where the high-resolution part of the image is along the optical axis, we can use lenses that have lower quality off the optical axis, as long as the optical quality falls off slower than the resolution of space-variant image. This enables us to make a smaller camera than is available commercially.

In addition, most industrial video cameras output a standard analog RS-170 video signal which entails further electronics to digitize the data and grab the frame. Our camera readout circuitry avoids this entire step by digitizing the sensor data and putting it directly on the processor's data bus.

We designed the camera to fulfill several design criteria. They are:

- The camera head had to be very small and light in order to be actuated by the SPM.
- In order to minimize the associated electronics, the camera should be read out in a digital form suitable for input directly to a computer. By avoiding a standard analog video signal, the system is greatly simplified.
- It must have sufficient field of view and resolution to satisfy our benchmark's requirements. Our chosen problem is to track a moving car and read the license plate on it. The fixed field of view lens does not have sufficient resolution to read the license plate at large enough distances. The zoom lens, however, can be used to track the car in the wide angle position, and read the license plate in the telephoto position.
- Because we use the camera to generate space-variant images, it is not necessary for the lenses we use to maintain high optical quality across the entire field.

In this chapter, we discuss the details of the optics, the electronic readout of the camera, and a fast algorithm to map the uniform image to a logmap image.

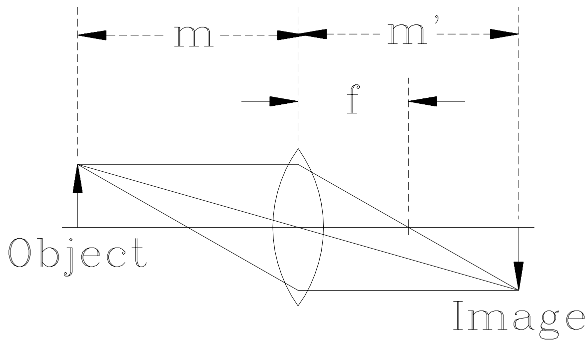


Figure 2.4: Illustration of imaging an object with a simple biconvex lens.

2.2 Camera optics

The optics of a camera consists of the elements necessary to focus the light from an external scene to make an image on a sensor. In our case, this includes the lenses to focus the light, the lens housing to hold the lenses and an actuator to move the zoom lens. Some of the issues in designing such a system are the field of view of the lens, optical quality across the field of view, availability of lenses, and the zoom mechanism. In this section, we examine these issues in the design of our camera.

One of the simplest optics designs consists of a single biconvex lens. Such a system, depicted in Figure 2.4, produces an inverted image at a distance

$$m' = (mf)/(m - f) \tag{13}$$

from the lens [28, P. 46] where f is the focal length of the lens, m is the object distance and m' is the image distance. Since the sensing element is a fixed distance from the lens, we would like m' to be constant. If m is large compared to f, then m' will not vary by much as m varies. By building a camera with a small enough focal length, we can avoid including a focusing mechanism as the image will always be in focus.

For this or any optics system, the system must be designed so that in coordination with the sensor being used, we obtain the desired field of view. The best way to determine the field of view of a camera is to simulate the path of light rays from outside the camera through the lenses to the sensor. Because the surfaces and index of refraction of each lens element is known, it is possible to compute the refractive path of each ray across each surface. This is a process called ray-tracing. Using a commercial ray-tracing package, we determined that a 4 mm focal length lens with our sensor would yield a 33^o field of view, which is about what we wanted.

Unfortunately, such a simple optical system has severe chromatic aberrations. This means that the focal length of the lens is different for different wavelengths of light resulting in different image planes for different colors. For us, this would mean an out of focus image since real-world scenes are not monochromatic.

Chromatic aberrations can be corrected by adding more lens elements [28, P. 157–163]. The design of such systems is rather complicated, and rather than reinvent the wheel, we use such a lens system from the eyepiece of a telescope. This lens system, known as an orthoscopic lens assembly, makes up the entirety of our fixed field of view lens. The mechanical drawing of the camera with these lenses is shown in Figure 2.5. This camera head is $8 \times 8 \times 10$ mm and weighs less than 6 grams.

This camera was meant to be used to test our system, and in that it was successful. We were able to test the pan-tilt actuation of the camera, the electronic readout, and image processing in the space-variant domain. The field of view and resolution are not high enough, however, to solve our license plate

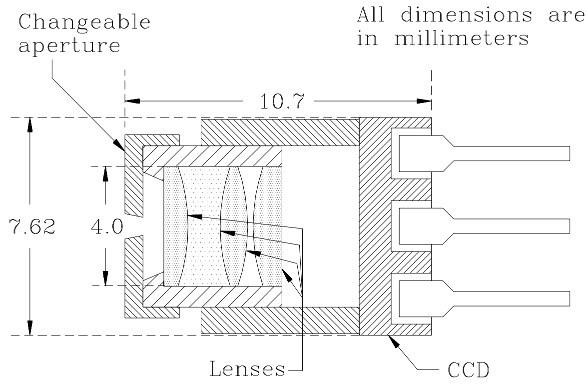


Figure 2.5: Mechanical drawing of the fixed field of view camera. The 4 element lens is a 4 mm orthoscopic lens that is corrected for chromatic aberrations.

problem well. When the license plate is far enough away to track its motion (i.e., when the entire license plate is in the field of view), there is barely enough resolution to read the characters on it. We could solve this problem by using either a higher resolution sensor or having a lens with controllable field of view. We chose the latter.

2.2.1 Zoom lens

We designed a zoom lens capable of operating between a 4 and 13 mm focal length. This corresponds to a 33^o and a 10^o field of view, respectively, which is sufficient to solve our license plate problem. The lens is actuated between positions with a *solenoid*. Some of the issues in designing this zoom lens are its focusing, the lens availability, the brightness across the field of view, and the mechanical assembly and actuation.

The simplest zoom lens can be built by just moving a single lens, such as the one depicted in Figure 2.5, between the object and the image. The image will change magnification, but the focal plane of the image will move significantly, according to Equation 13. The situation can be improved by adding two lenses (Figure 2.6). Here, as the middle negative lens moves between the two outer positive lenses, the overall focal length changes, and while the focal plane still moves, its motion is greatly reduced. The focal plane is at the same position for exactly two different zoom positions. This can be adjusted so that the two positions are the minimum and maximum zoom, which is what we use since we can actuate it only to these two positions. This type of zoom lens system is examined in great detail by Bergstein [9,10]. Our zoom lens system is based on that in Figure 2.6 with the modification that the last lens is replaced with the orthoscopic lens of the fixed field of view camera to improve the chromatic aberrations. In addition, an internal aperture is added to limit light reflecting off the walls of the lens housing.

In designing the zoom lens, we were limited by the availability of commercial lenses. Because of the extreme difficulty of finding very small lenses meeting our specifications, we had to build the zoom lens somewhat larger than we had anticipated. One solution to this problem is the custom manufacturing of small lenses ourselves. We are able to make plano-concave and biconcave lenses using optical quality epoxy with a release agent so the epoxy does not stick to the mold. We pour the epoxy into brass tubing and place metal spheres of the appropriate diameter on the ends of the tubing. The epoxy dries and we remove the spheres, leaving a lens inside the tubing. We have made some prototype lenses using this technique and are in the process of making a new zoom lens assembly with them.

One problem with multiple element lenses such as the zoom lens is internal reflection of light off of the inside of the lens housing. This occurs most often with light entering the lens assembly at large

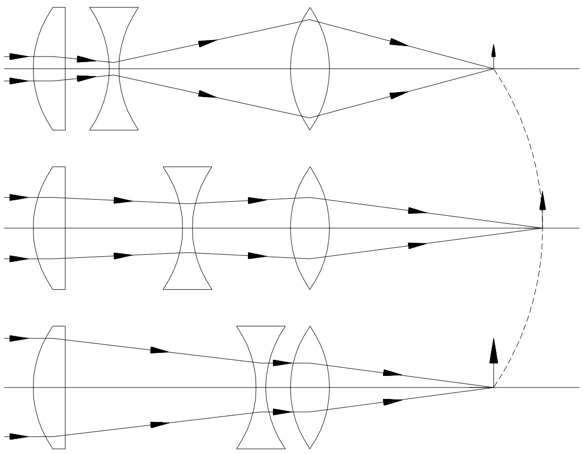


Figure 2.6: Illustration of zoom lens. The image plane moves as the magnification of the lens changes. (Adapted from [51, P. 275])

angles. This can be greatly reduced by inserting an aperture stop in the lens between lens elements. This is a disk of metal that blocks all light rays very near the edge of the lenses where most of the unwanted light is.

The aperture stop increases the vignetting of the lens, which is the reduced illumination of the image plane in the periphery of the image. This is because light entering the lens assembly at large angles gets cut off by the stop that otherwise would end up in the periphery of the image. In our case, the vignetting is not very severe, and can by compensated for in software.

The mechanical assembly holding the lenses had to be designed to allow a very small force to actuate the central lens element while not allowing light to leak into the system. At the same time, the lenses must all be adjustable so the system can be focused. A mechanical drawing of the assembly we designed is in Figure 2.7. We mounted the two outer lenses on threads so they are easily focusable. The central lens slides inside the lens housing, attached to two pins that connect to a a small cylinder outside the housing. This lens is actuated by applying current to a coil of wire wrapped around it. There are two press-fit rings that act as stops to control the limits of motion of the central lens. Two pieces of polarized plastic are put in the lens assembly, one fixed and one rotatable, to control the light intensisty. A thin cylindrical baffle fits over the entire assembly to block light from leaking in the slots. The zoom lens is pictured with and without the baffle in Figure 2.3.

The zoom lens is actuated with a small solenoid. A coil of wire is wound around the small cylindrical sheath attached to the zoom lens by pins. Because the camera is mounted directly against the pole of the permanent magnet in the rotor of the SPM, applying current to the coil results in a force on the zoom lens in either direction depending on the polarity of the current. Although the zoom lens is moderately well-focused throughout its range, we only have the capability of actuating it to two positions, one at each end of its range of motion.

Images taken using the zoom lens are in Figure 2.8.

2.3 Image sensor

In designing our camera, we had to choose between two image sensor technologies, CMOS and CCD. They are both silicon-based VLSI technologies, and both are able to transduce light to voltages. In this section, we discuss the basics of these technologies and explain the details of the hardware and software

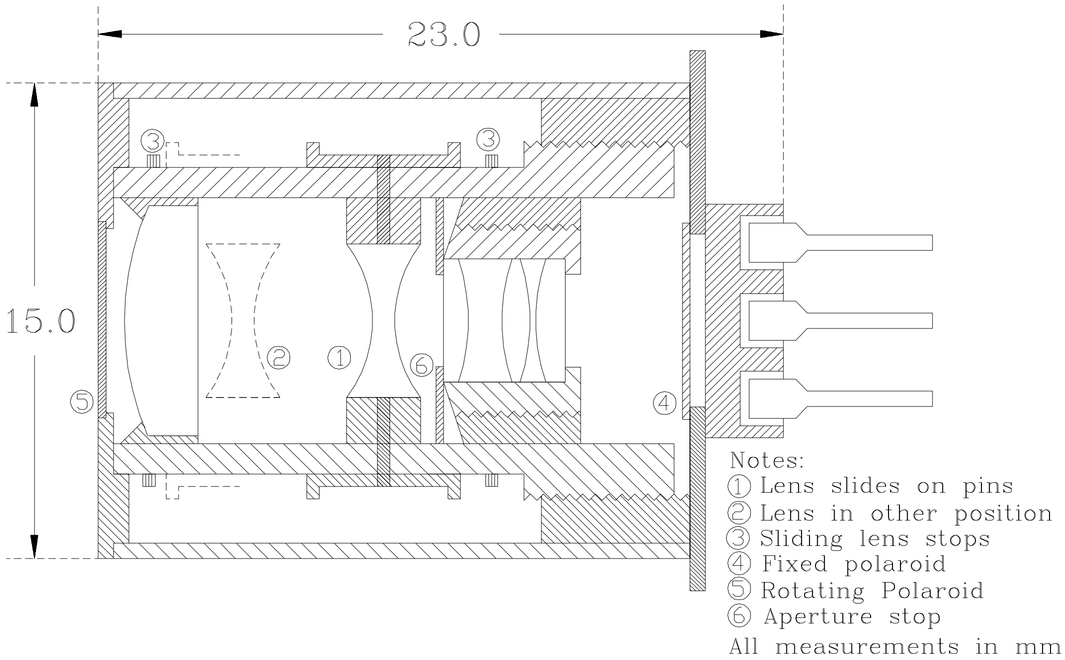


Figure 2.7: Mechanical drawing of the zoom camera.

we developed to use the sensor.

2.3.1 Sensor technologies

CMOS image sensors are based on bipolar junction transistors (BJT's) which are inherently light sensitive (Figure 2.9). If the base of a BJT is not connected, the current through the transistor will be proportional to the light incident on it. We can increase the dynamic range of the sensor by taking the logarithm of this current. We take this logarithm by connecting two field effect transistors to the BJT [35]. The chip we are making with Synaptics uses this approach.

A CMOS image sensor consists of an array of VBJT's with an addressing scheme that allows each transistor to be accessed directly. The result of this scheme is that an image frame may be read from the sensor by scanning through each transistor sequentially. The next frame can be read directly without delay. The maximum frame rate is determined by the speed with which each transistor can be addressed. CMOS image sensors are usually fast and easy to use, but they are relatively large. The smallest pixel in our current design is 10 microns.

CCD image sensors, on the other hand are denser with pixel sizes down to 5 microns, resulting in their more common usage. They work by accumulating charge, proportional to the incident light, in an electronic "bucket". Each bucket must accumulate (or integrate) charge before it can be read out. This results in a different usage than CMOS sensors. Between each frame read out, the CCD must integrate light. Thus, the maximum frame rate is determined not only by the readout speed (or scan time), but also by the integration time.

CMOS sensors are usually capable of addressing each pixel individually. CCD sensors, however, work by shifting the contents of each row of pixels vertically through the other pixels to the edge of the array where they are either read out through a horizontal shift register, or stored in a frame buffer on chip for future readout. During this shifting process, as the image is passed through every pixel in its column,

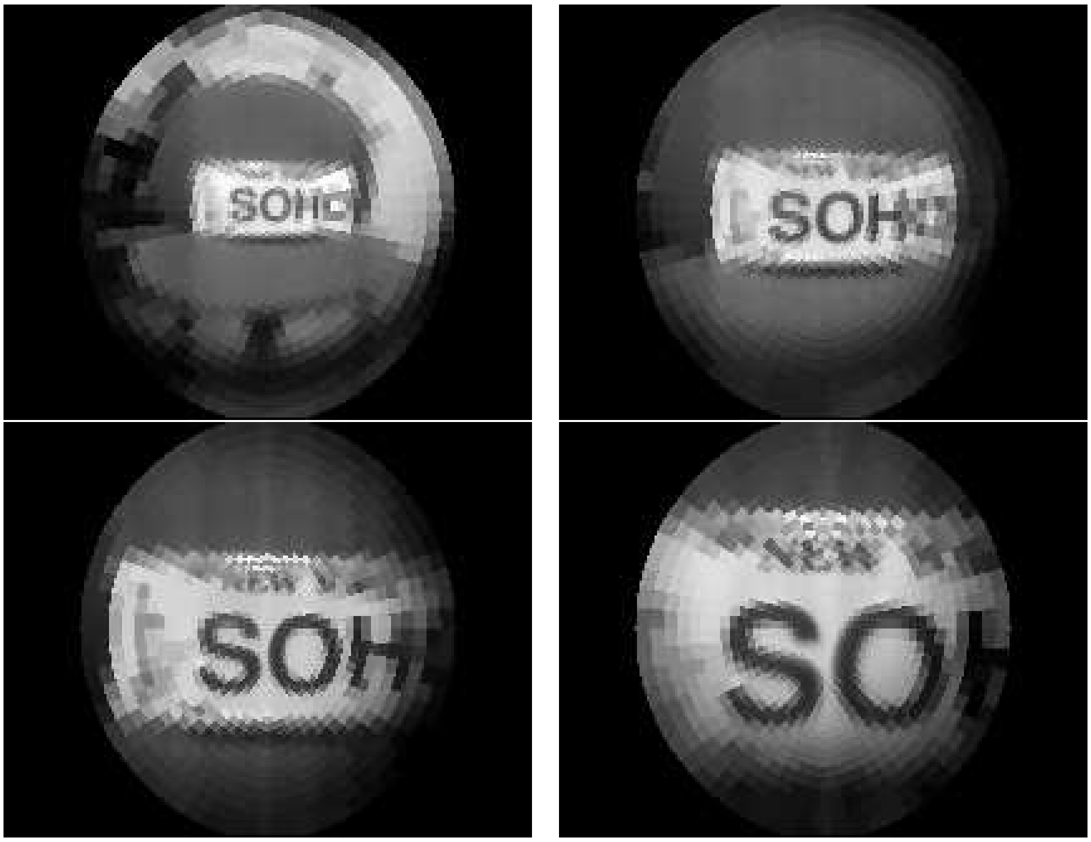


Figure 2.8: The same scene imaged with the zoom lens in four different positions (manually actuated).

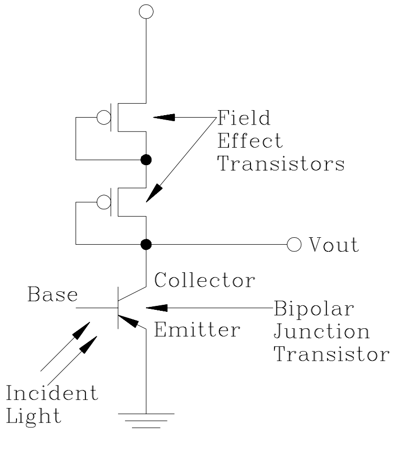


Figure 2.9: A vertical bipolar juction transistor hooked up with two field effect transistors to produce a photo cell with a logarithmic output.

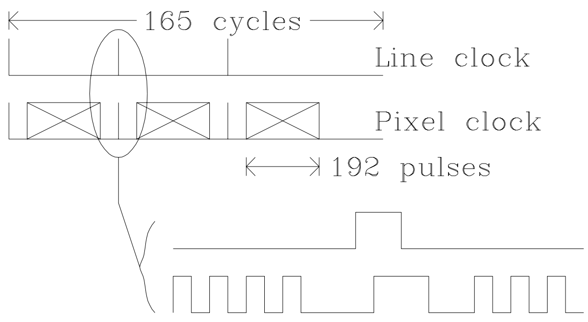


Figure 2.10: Timing signals used to control the CCD image sensor.

the sensor continues to integrate light. This results in *vertical smearing* where each pixel has an influence on every other pixel in its column, where the amount of influence is proportional to the ratio of scan time to integration time. Because of this, some CCD sensor chips have a frame buffer that is not light sensitive to which the image is shifted very quickly. The image can then be read from this frame buffer relatively slowly without affecting image quality. In this case, there is still some vertical smearing, but it is greatly reduced because the transfer time to the frame buffer is very small.

2.3.2 Timing signals

A CCD needs many different signals to control its readout and light integration. As already stated, each pixel in the CCD consists of an electronic "bucket" accumulating charge. If the light intensity is too great, or the integration time is too long, the bucket can overflow into its neighbors. Thus, one pixel can influence its neighbors. This is called blooming. Many CCD sensors have an antiblooming control where the top portion of the bucket is discarded at regular intervals. This eliminates blooming for all but the most intense illuminations, or longest integration times.

We chose the smallest rectangular image sensing device commercially available, which is a CCD sensor without an on-chip frame buffer. It is the Texas Instruments TC211 device and contains an array of 165×192 pixels in an 8×8 mm 6 pin DIP package. The six pins of the chip are power, ground, line clock (IAG), pixel clock (SRG), anti-blooming clock (ABG), and output. Each frame is read out by putting one pulse on SRG for each pixel, and reading the voltage at the output. At the end of each line, a pulse is put on IAG. After a frame is completely read out, a clock is put on ABG while the chip integrates light after which the next frame is read out in the same manner. These timing signals are described in detail in Figure 2.10.

We developed a printed circuit board that in coordination with a processor, generates the signals for, and digitizes the output from the CCD. It does not generate the optional anti-blooming clock. The schematic diagram for the circuit is in Figure 2.11. It reads the CCD at a maximum pixel rate of 2.5 MHz which corresponds to 75 frames per second. Because the CCD needs an integration time that is usually at least equal to the scan time, the usable frame rate is no more than 32 frames per second.

The circuit reads the CCD one row at a time, storing the digitized data in a FIFO buffer that the processor then reads. It has two inputs to accomplish this, RESET and START. RESET initializes the circuit, clearing the FIFO. START starts the circuit readout of one row of the CCD into the FIFO. Because the FIFO is dual-ported, after one pixel time, the processor can start reading that row from the FIFO. The CCD automatically shifts each row down one row when IAG is pulsed, so each time the processor reads a row, it gets the next row from the CCD. The circuit does not do anything at the frame level. Rather, it is up to the processor to read the correct number of rows and then let the CCD integrate for a fixed amount of time before reading the next frame. A description of the system architecture and

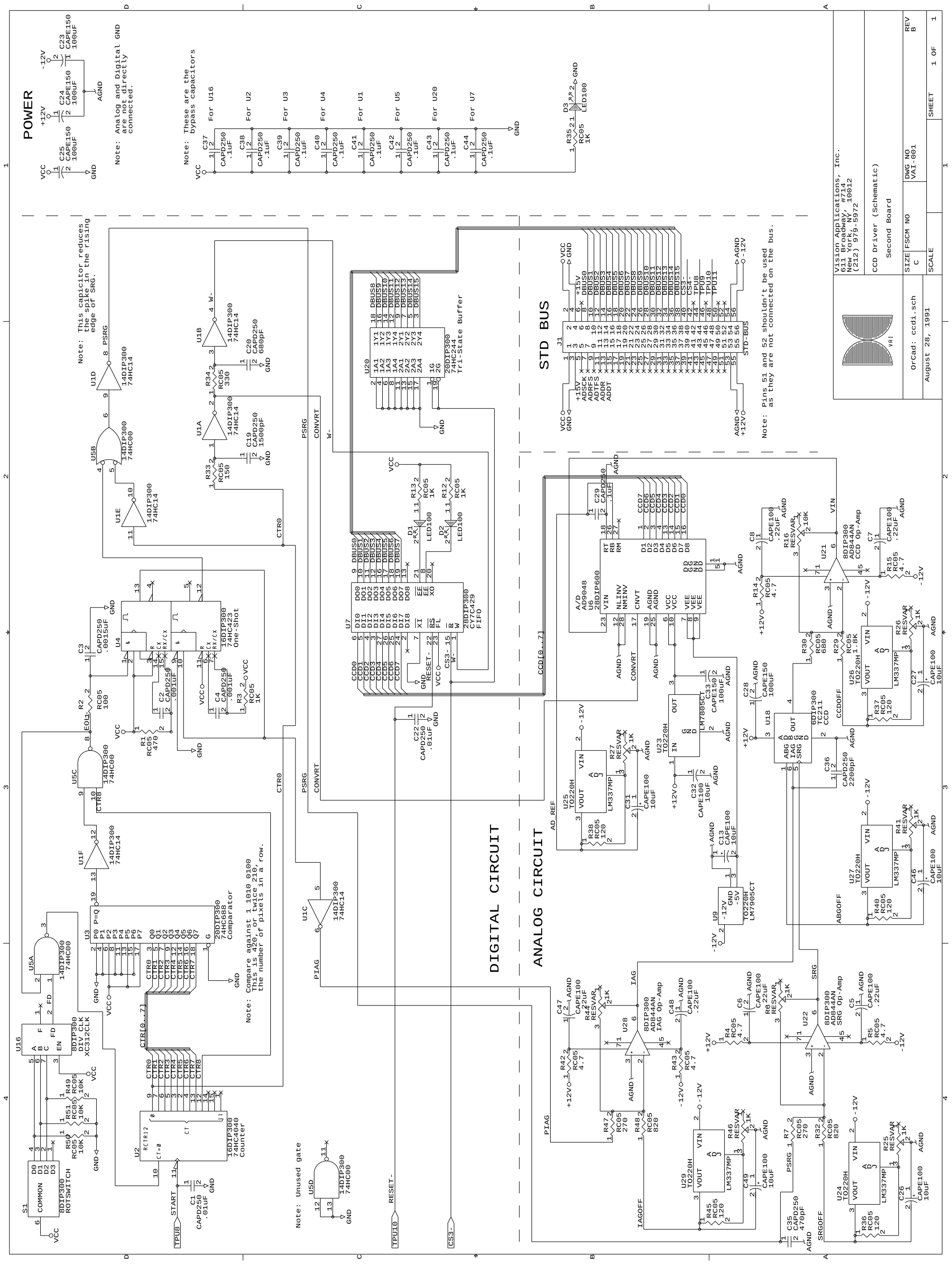


Figure 2.11: Schematic diagram for CCD driver board.

the processors used to read from the CCD driver board can be found in Chapter 5.

The circuit is divided into a digital and an analog section. The above mentioned part of the circuit is all digital. The analog part converts the signals to the necessary levels, and amplifies and digitizes the CCD output to be stored in the FIFO.

2.3.3 Vertical smearing

The CCD we use has a significant image degradation problem called vertical smearing. The processor must read the CCD as quickly as possible because the CCD continues to integrate light as the image is shifted out of the chip. Each pixel that is read out actually consists of itself plus a fraction of all other pixels in that column. The vertical smearing increases with the ratio of scan time to integration time.

We model the vertical smearing exactly for static scenes with a linear algebraic system, enabling us to elminate the smearing in software. We use the definitions of TV and logmap images of Section 1.3. Because each pixel read out is a linear combination of the original pixel and all the other pixels in that column, the original pixel value can be retrieved by solving a set of linear equations. Let us call the pixel that is read out from the CCD $I'(i,j)$ and the actual pixel value before it is read out $I(i,j)$ where $i \in \{0,\ldots,\mathsf{NROWS}-1\}$ and $j \in \{0,\ldots,\mathsf{NCOLS}-1\}$. Then for each column, j, we have a set of pixels, $I'(i,j)$, and want to calculate the actual values, $I(i,j)$. We say the integration to scan-time ratio is $\mathsf{INT\_SCAN} : 1$. This means that a read out pixel, $I'(i,j)$, contains $\mathsf{INT\_SCAN}/(\mathsf{INT\_SCAN}+1)$ of the actual pixel value, $I(i,j)$, and $1/(\mathsf{INT\_SCAN}+1)$ of all the other pixels in the column. We can write this as

$$I'(i,j) = \frac{\mathsf{INT\_SCAN}}{\mathsf{INT\_SCAN}+1} I(i,j) + \frac{1}{\mathsf{NROWS}(\mathsf{INT\_SCAN}+1)} \sum_{k=0}^{\mathsf{NROWS}-1} I(k,j) \tag{14}$$

We must solve these NROWS linear equations in NROWS unknowns for the actual pixel values, $I(i,j)$. This must be done separately for each column. This can be done off-line and then calculating each actual pixel, $I(i,j)$, would involve NROWS multiplications and NROWS additions. Because we are interested in logmap images that are derived from the original TV image, we do not want to desmear the original TV image. Instead, we examine the vertical smearing problem in the logmap domain where we find an approximate solution.

In logmap images, each pixel is the average of many TV image pixels and these pixels come from multiple columns. Again, we want to find the actual logmap pixel value (which is the average of actual TV pixel values) from the read out logmap pixel values (which are the averages of readout TV pixel values), and we want to do this without referring to the TV pixels themselves. We call a read out logmap pixel, $L'(u,v)$, and an actual logmap pixel, $L(u,v)$, where $u \in \{0,\ldots,\mathsf{NSPOKES}-1\}$ and $v \in \{0,\ldots,\mathsf{NRINGS}-1\}$. These are defined by Equation 2 of Section 1.3. Now we can solve for the actual value of a logmap pixel analagously to rectangular pixels. From Equation 14, we write

$$\begin{aligned}
L'(u,v) &= \frac{1}{a(u,v)} \sum_{i,j} \Bigg(\frac{\mathsf{INT_SCAN}}{\mathsf{INT_SCAN}+1} I(i,j) + \\
&\quad \frac{1}{\mathsf{NROWS}(\mathsf{INT_SCAN}+1)} \sum_{k=0}^{\mathsf{NROWS}-1} I(k,j) \,| \\
&\quad S(i,j) = u \text{ and } R(i,j) = v \;\Bigg) &(15)\\
L'(u,v) &= \frac{\mathsf{INT_SCAN}}{(\mathsf{INT_SCAN}+1)a(u,v)} \sum_{i,j} I(i,j) \,|\, (S(i,j) = u \text{ and } R(i,j) = v) \\
&\quad + \frac{1}{\mathsf{NROWS}(\mathsf{INT_SCAN}+1)a(u,v)} \sum_{i,j} \sum_{k=0}^{\mathsf{NROWS}-1} I(k,j) \,| \\
&\quad (S(i,j) = u \text{ and } R(i,j) = v) &(16)
\end{aligned}$$

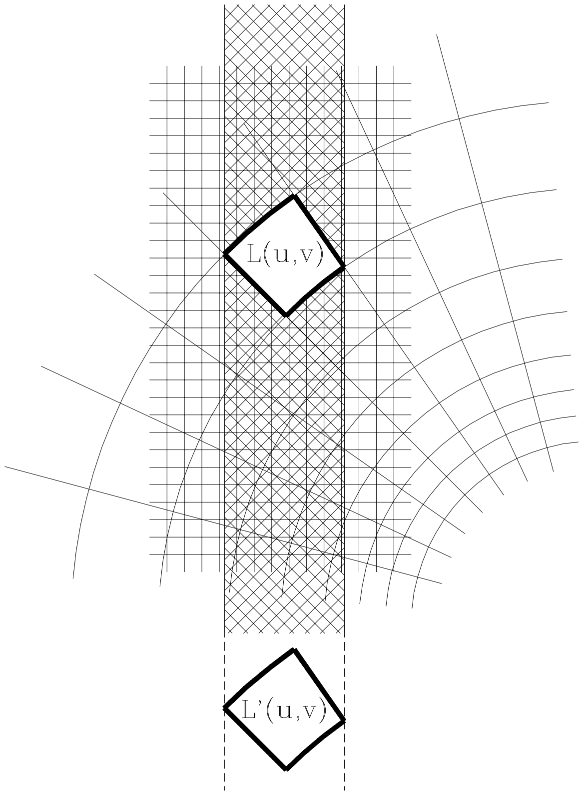


Figure 2.12: Elimination of vertical smearing. Actual logmap pixel $L(u,v)$ is calculated from read out logmap pixel $L'(u,v)$ and a weighted sum of all other logmap pixels sharing TV pixels in the same columns.

$$\begin{aligned} L'(u,v) &= \frac{\mathsf{INT\_SCAN}}{\mathsf{INT\_SCAN}+1} L(u,v) + \\ & \frac{1}{\mathsf{NROWS}(\mathsf{INT\_SCAN}+1)a(u,v)} \sum_{i,j} \sum_{k=0}^{\mathsf{NROWS}-1} I(k,j) \mid \\ & (S(i,j)=u \text{ and } R(i,j)=v) \end{aligned} \tag{17}$$

The second term of Equation 17 represents the sum of TV pixels in the same column as the logmap pixel $L(u,v)$. This is represented in Figure 2.12. This can not be calculated in terms of logmap pixels exactly because some logmap pixels have only some of their TV pixels in the same column as $L(u,v)$. In addition, each TV column is weighted differently depending on the number of TV pixels in a column of $L(u,v)$.

We approximate Equation 17 using only logmap pixels. We use weighted logmap pixels where the weight of each pixel is the total number of TV pixels from that logmap pixel corresponding to TV pixels in $L(u,v)$. We define the weights, $W(u,v,w,x)$, for each logmap pixel, $L(u,v)$, using the total weight, $W_{TOT}(u,v)$, to scale each logmap pixel as defined in Equations 18 and 20. Note that the definition for $W_{TOT}(u,v)$ counts only the TV pixels in a column that map to $L(u,v)$.

$$W_{TOT}(u,v) = \sum_{i,j} \sum_{k=0}^{\mathsf{NROWS}-1} 1 \mid S(i,j) = u \text{ and } R(i,j) = v \text{ and} \tag{18}$$

$$S(k,j) \neq \text{invalid} \tag{19}$$

$$W(u,v,w,x) = \sum_{i,j,k} 1 \mid S(i,j) = u \text{ and } R(i,j) = v \text{ and}$$

$$S(k,j) = w \text{ and } R(k,j) = x \tag{20}$$

We can rewrite Equation 17 using this approximation to get

$$L'(u,v) = \frac{\mathsf{INT\_SCAN}}{\mathsf{INT\_SCAN}+1} L(u,v) + \frac{1}{(\mathsf{INT\_SCAN}+1) W_{TOT}(u,v)} \times$$

$$\sum_{k=0}^{\mathsf{NROWS}-1} \sum_{l=0}^{\mathsf{NCOLS}-1} W(u,v,k,l) \cdot L(k,l) \tag{21}$$

The result of this approximation is that pixels are smeared horizontally no more than the width of any logmap pixel sharing TV pixels in the same column. This is because logmap pixels sharing TV pixels in the same column may extend to other columns not covered by the original logmap pixel. This, however, is a large improvement over the original vertical smearing that extended over the full height of each column.

The implementation of this desmearing approach is complicated by the use of the logmap. In the rectangular case, each column of TV pixels can be solved independently because each TV pixel in a column depends only on other TV pixels in the same column. Logmap images, on the other hand, have no columns. Each logmap pixel depends on the set of logmap pixels sharing TV pixels in the same column as itself. The other logmap pixels in this set, however, do not generally depend on the same set of pixels. This means that there is no way to solve the desmearing problem for any subset of the image. Instead, the entire image must be desmeared in one step. Referring to Equation 21, this involves solving NPIXELS linear equations with NPIXELS unknowns where NPIXELS is the number of pixels in the logmap image.

This at first may seem a daunting task, but the matrix to solve is sparse. For our logmap, there are $1,376$ pixels. The number of pixels that each pixel depends on varies with the particular pixel, but varies from 12 to 246 with a mean of 86. This means that an average row in the matrix of linear equations will have only 86/1376 or 6% non-zero entries.

An image can be desmeared by taking advantage of the sparseness of the matrix and by doing most of the computation off-line. Equation 21 can be rewritten as a matrix problem, $\mathcal{A}\mathcal{X} = b$, where $\mathcal{X}$ is the unknown actual logmap pixel values ($L(u,v)$ in a 1D vector form), b is the known read-out logmap pixel values (L' in a 1D vector form), and $\mathcal{A}$ is the matrix of coefficients multiplying $\mathcal{X}$. We first invert $\mathcal{A}$ off-line using sparse matrix inversion techniques ([39, P. 72–81]). To desmear a given image, we vectorize it into b, then just compute the result, $\mathcal{X} = \mathcal{A}^{-1}b$. This still requires a large amount of computation. Fortunately, $\mathcal{A}^{-1}$ is also sparse and thus b only needs to be multiplied with non-zero entries of $\mathcal{A}^{-1}$. This can be sped up further by eliminating near-zero values of $\mathcal{A}^{-1}$. Figure 2.13 shows the result of applying these desmearing techniques to an image taken with our camera. Results are shown using the full $\mathcal{A}^{-1}$, using only values of $\mathcal{A}^{-1}$ whose absolute values are greater than 0.001, and greater than 0.01. Severe degradation along the vertical meridian results for the last case. The vertical meridian is the most sensitive area because here, there are the most pixels in a column. Because each pixel depends on so many other pixels, their weights are small. Results showing timing and the number of elements in $\mathcal{A}^{-1}$ used are shown here where the execution time is for a SUN Sparc 2.

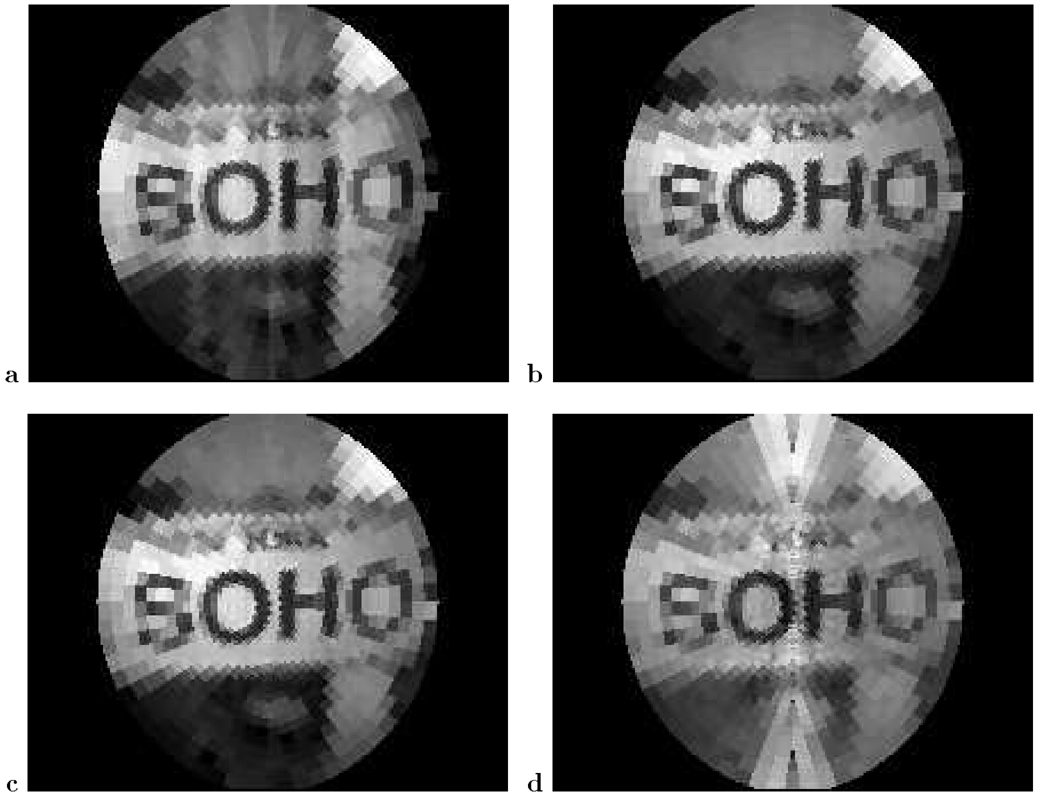


Figure 2.13: **a)** Image from miniature camera with vertical smearing; **b)** desmeared image; **c)** desmeared image using only $|\mathcal{A}^{-1}| > 0.001$; **d)** using only $|\mathcal{A}^{-1}| > 0.01$.

$\mathcal{A}^{-1}$ cutoff	% of $\mathcal{A}^{-1}$ used	Execution time
—	30%	1.36 seconds
.001	8%	0.58 seconds
.01	3%	0.19 seconds

2.3.4 Electrical noise

One of the biggest difficulties in obtaining a high quality image from the sensor is electrical noise. Noise comes from two main sources, electromagnetic interference (EMI) and noise in the readout circuit due to high frequency digital circuitry. The noise in the circuit can be greatly reduced through standard noise reduction techniques [25], such as using separate power supplies for the digital and analog parts of the circuit, using ground planes in the circuit, and placing related parts near each other.

EMI is especially bad in our case because the camera is mounted in the SPM approximately 20 cm from the readout circuit. To make matters worse, extremely thin wires (40 gauge) must be used to connect to the CCD in the SPM. This is because when the SPM moves the camera, it must also bend the wires attached to the camera. The connecting wires must be highly compliant in order to maximize the accuracy of the motor. The very thin wire is used for approximately 5 cm. We reduced the EMI by using twisted pair cables for the connections to the CCD. This entails twisting each signal wire with a ground wire.

In addition to these design techniques, we designed and had manufactured a printed circuit board for the readout circuit. This board substantially reduced the noise compared with the wire wrap boards we were using before.

2.3.5 Future Designs

In the future, the camera would be most improved by using a higher resolution image sensor. Because a higher resolution sensor will very likely be larger physically, the optics would have to be redesigned because the field of view and the off-axis optical quality would have to be increased. This would also necessitate using more computational power to compute the larger logmap.

The zoom lens could be further miniaturized. Its current large size is due to the unavailability of certain small lenses. In the future, we expect to be able to manufacture any lens we need ourselves with epoxy as discussed. In addition, the lens housing could be miniaturized. We made the lens assembly with threads for all the lenses so it would be easily adjustable. By replacing the threads with press fit joints, the assembly could be made substantially smaller.

The image quality could be improved by using a differential transmission scheme for the CCD output signal. A surface mount differential output op-amp would be connected directly to the CCD. This makes positive and negative outputs that would be twisted together. The circuit board would then convert this back to a regular signal for use by the rest of the circuit. The advantage of this technique is that both the positive and negative signals are affected the same way by EMI and when the signals are combined back together, the noise is subtracted out.

2.4 Fast logmap computation

The final step in using the miniature camera is to map the uniform output image into a logmap image using the look-up-tables $S(i,j)$ and $R(i,j)$ of Section 1.3. This can be done by averaging the TV pixels that map to each logmap pixel as specified by $S(i,j)$ and $R(i,j)$. This approach, however, is relatively slow requiring several instructions per TV pixel.

We have developed a fast algorithm to compute the logmap based on a run-length mapping[1]. This algorithm avoids accessing the look-up-tables for each pixel and it executes with an average of just over one instruction per TV pixel. The $S(i,j)$ and $R(i,j)$ look-up-tables are first run-length encoded off-line with the algorithm **run_length_encode**. The CCD image is then mapped, line by line, as it is read out into the logmap image with the algorithm **fastmap**.

The algorithm **run_length_encode** analyzes $S(i,j)$ and $R(i,j)$ in raster order to find the *runs*, or sequences of length ℓ in which

$$S(i,j) = \cdots = S(i,j+\ell-1)$$

and

$$R(i,j) = \cdots = R(i,j+\ell-1).$$

We partition the raster scan of the TV image into a sequence of NRUNS runs where run_len[k] is the length of the kth run and run_addr[k] is the address of the logmap pixel of the kth run.

The algorithm **fastmap** first zeros the logmap. It then goes through the TV image in raster order and through the runs in linear order. For each run, k, it adds run_len[k] TV pixels to the logmap pixel indexed by run_addr[k]. After the entire image has been mapped, the resulting logmap image must be normalized. For each logmap pixel, $L(u,v)$, we set $L(u,v) = L(u,v)/a(u,v)$. **fastmap** shows the pixels in each run being summed in a loop. However, in practice we avoid the overhead associated with each loop iteration by unrolling the loops. We call a routine, $\mathbf{map}_n$ which consists of n inline summing statements of the form `*lm_addr += I(tv_index++)`.

[1]Patent pending.

The number of runs, NRUNS, depends on the mapping function given by the look-up-tables $S(i,j)$ and $R(i,j)$. For a wide variety of interesting mappings, such as the logmap, $\mathsf{NRUNS} \ll \mathsf{NROWS} \times \mathsf{NCOLS}$, and $\mathsf{NSPOKES} \times \mathsf{NRINGS} \ll \mathsf{NROWS} \times \mathsf{NCOLS}$. Let us assume that the statements in **fastmap** that add the TV pixel to the logmap pixel and increment the TV pixel address takes one assembly language instruction. Let us also assume that we either unroll the loop computing each run (or use a DSP processor with zero-overhead loops) so that there is no overhead during each run. If the overhead between each run takes b instructions, then the mapping will take $\mathsf{NROWS} \times \mathsf{NCOLS} + b \times \mathsf{NRUNS} + 2 \times \mathsf{NSPOKES} \times \mathsf{NRINGS}$ instructions, or $1 + \frac{b \times \mathsf{NRUNS}}{\mathsf{NROWS} \times \mathsf{NCOLS}} + \frac{2 \times \mathsf{NSPOKES} \times \mathsf{NRINGS}}{\mathsf{NROWS} \times \mathsf{NCOLS}}$ instructions per TV pixel.

Our system uses a TV image of $165 \times 192 = 34,650$ pixels and there are $6,074$ runs. The logmap image is $34 \times 56 = 1904$ pixels, and the number of "overhead" instructions between each run, b, is 4. Then the mapping takes 1.81 instructions per TV pixel.

Algorithm: **run_length_encode**
Function: This creates the run length and address tables for a mapping based on the spoke and ring look-up-tables.
Input: $S(i,j)$ and $R(i,j)$ look-up-tables
Output: Run length table, run_len and run address table, run_addr.

```
length ← 0
run_index ← 0
old_address ← -1
For each TV pixel, (i,j), in raster order
    If ((j = NCOLS-1) OR (R(i,j) = invalid)) Then
        address ← -1
        spoke ← 0              { a(spoke,ring) must equal 0 }
        ring ← NSPOKES/2
    Else
        address ← S(i,j) × NRINGS +R(i,j)
        spoke ← S(i,j)
        ring ← R(i,j)
    End If
    If (((j > 0) AND (address ≠ old_address)) OR (j = NCOLS-1)) Then
        run_len[run_index] ← length
        run_addr[run_index] ← old_spoke × NRINGS + old_ring
        run_index ← run_index + 1
        length ← 1
    Else
        length ← length + 1
    End If
    old_address ← address
    old_spoke ← spoke
    old_ring ← ring
End For
NRUNS ← run_index
```

Algorithm: **fastmap**

Function: This maps the uniform TV image, $I(i, j)$, to the logmap image, $L(u, v)$. It uses the run length and address tables computed by **run_length_encode.**

Input: run_len and run_addr run tables
TV image, $I(i, j)$

Output: Logmap image, $L(u, v)$

```
tv_index ← 0                      { The TV image, I, is accessed in raster order with the index,
                                  tv_index. }
For each pixel p ∈ L              { Zero logmap }
    L(p) ← 0
End For
For each run, k
    lm_addr ← address_of(L) + run_addr[k]
    For l = 0 to (run_len[k] − 1)
                                  { The next two lines can be written with one C instruction,
                                  *lm_addr += I(tv_index++). Note that the '*lm_addr' no-
                                  tation denotes the memory location referenced by lm_addr }
        *lm_addr ← *lm_addr + I(tv_index)
        tv_index ← tv_index + 1
    End For
End For
For each pixel p ∈ L              { Normalize logmap }
    L(p) ← L(p) / a(p)
End For
```

Chapter 3

Platform Pantilt

3.1 Introduction

Once we have a miniature camera for acquiring space-variant images, we need a method of actuating the camera. We would like a pan-tilt mechanism that is accurate, fast, small, inexpensive and has low power requirements.

We have constructed two new actuators meeting these requirements. The first is called the Platform Pantilt, a pan-tilt actuator based on two linear stepper motors (Figure 3.1) described in this chapter. The second actuator, the Spherical Pointing Motor (SPM), incorporates both pan and tilt into a single two degree of freedom motor (Figure 4.1). It consists of three orthogonal motor windings in a permanent magnetic field, configured to move a small camera attached to a gimbal. It is an absolute positioning device and is run completely open loop. It is described in Chapter 4. Both of these actuator designs are different than traditional pan-tilt actuators in that they avoid the inefficiencies involved with one motor moving another motor.

Pan-tilt mechanisms have been a source of inspiration and frustration to computer vision researchers. One source of inspiration is nature, since humans and animals rely on their pan-tilt apparatus to achieve wide field-of-view visual sensing.

The argument from nature is not by itself a compelling reason to build robot eyes that pan and tilt. The alternative to mechanical pan-tilt action is electronic scanning, i.e. computer control of a fixed set of cameras having a combined wide field-of-view. Selective attention can be implemented by a variety of addressing methods, for example the inverted pyramid of Burt [11,12]. Such "software pan-tilt" mechanisms are considerably more convenient and reliable than their motorized counterparts, but we argue that they ultimately are more expensive. A large sensor, which is required to efficiently use software pan-tilt, has a quadratically increasing pixel load. The alternative to providing the extra processor bandwidth and memory needed for such an approach is to physically move the sensor, as is the usual choice in "active-vision" approaches. It is our opinion that at the current state of technology, this is a considerably more economical approach.

Given today's inexpensive CCD camera arrays, electronic scanning may appear at first glance to be superior to the mechanical approach. Certainly such electronic pan and tilt is faster than mechanical pan and tilt. It also is more reliable, as there are no moving parts involved. But there are several arguments in favor of using mechanical actuation. They are:

cost We have shown at least one example in which the mechanical approach is less expensive. Our prototype SPM carries a camera having a 33 degree field of view. The actuator can pan and tilt the camera through a range of 60 degrees in each axis, resulting in an overall field of view of 93 degrees in each axis. To achieve the same result for the same CCD array in a nonmechanical configuration, we would have to build an array of 9 CCD sensors and lenses, arranged so that their

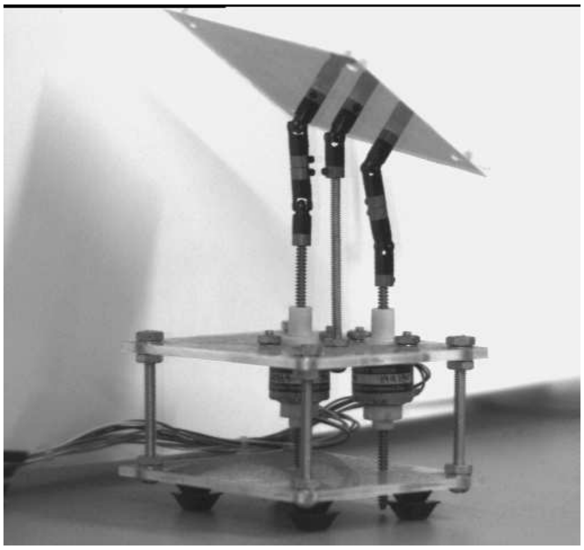


Figure 3.1: The Platform Pantilt. This stepper motor based actuator is capable of positioning a load of 80 grams, for example a small video camera. Dimensions are $7.5 \times 7.5 \times 12$ cm.

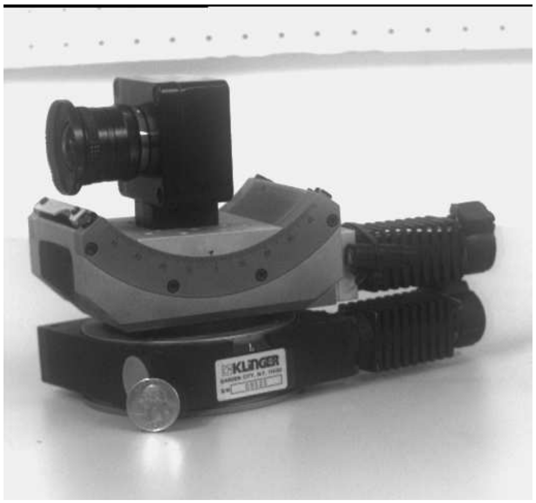
fields-of-view are adjacent. Even at the current low prices, 9 CCDs and lens assemblies cost an order of magnitude more than the parts in the SPM. The cost of CCDs increases exponentially with size. Of course, with either approach there is an electronic circuit driving the system. For the SPM, this circuit consists of a motor driver and PWM controller, plus the components necessary to digitize the camera signal. For the fixed array of cameras, the electronics for the camera signals must be repeated 9 times, so we claim that the cost of support electronics is comparable in either case.

registration A system with multiple image sensors would have to be very carefully calibrated in order to register the multiple images. Without knowing precisely how the external world mapped onto each sensor in relation to the other sensors, it would be very difficult if not impossible to use such a system.

optics Using space-variant images with the fovea at the center of the image, and thus in line with the optical axis of the lenses allowed us to miniaturize the camera in a way that would not otherwise have been possible. Our miniature camera has severe off-axis optical degradation. The low resolution in the sensor periphery, however, is lower than the optical resolution which makes an ideal match for our system. This would not work, however, in a system where the foveation point was changed electronically to points off the optical axis, because the optical resolution would be lower than the sensor resolution. New lenses would have to be designed that have high resolution in the periphery. It would be much harder to design such lenses that would be as small as our current lenses.

custom sensor We plan on replacing our uniform off-the-shelf sensor with a custom sensor that has an intrinsic logmap geometry. We have built our entire software platform with this in mind, never using the original uniform TV image. We will therefore be able to make this replacement without changing our software substantially. Once we have a true logmap sensor, it will no longer be possible to control the foveation point electronically.

The simplest and most obvious pan-tilt mechanism uses a two-stage motor-on-motor (MOM) design. The first motor turns the mechanism through one degree of freedom, usually pan, and the second through



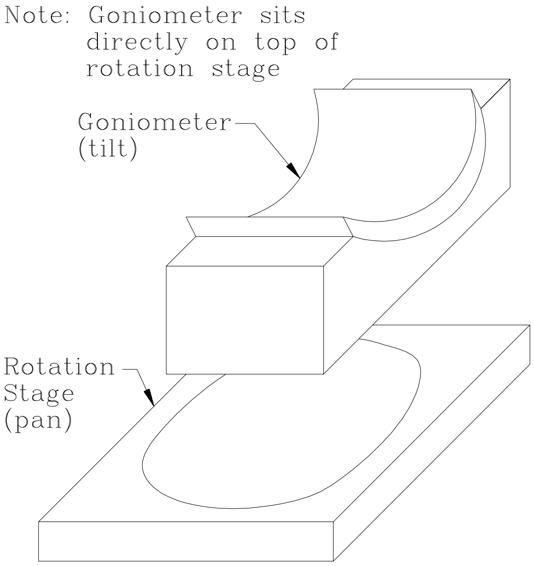


Figure 3.2: Photo and illustration of a traditional motor-on-motor design. Actuator from Klinger Scientific. Dimensions are $10 \times 11 \times 22$ cm (not including the camera or the rack-mounted controller/driver).

the other d.o.f, usually tilt. The second motor must be powerful enough to move the camera sensor. The first must move both the camera and the second motor. The MOM design therefore usually consists of one larger motor and one smaller one. Such a design is inefficient because, as we show, it is not necessary to carry one motor on top of another one. An example of such a design is the pan-tilt actuator available from Klinger Scientific (Figure 3.2).

The traditional solution to the inefficiencies of the MOM design is a parallel approach. This typically involves two motors which through some kind of linkage, actuate the rotor without having one motor moving the other. However, linkages are bulky and are difficult to make accurately. The Platform Pantilt is based on this method.

The Platform Pantilt moves a platform by raising and lowering two shafts with linear stepper motors that are fixed to the platform. The platform is attached to the base at a third point by a fixed shaft. All the shafts have some flexibility with different kinds of joints. This design is similar in some respects to the six degree of freedom Stewart platform [22,52,53]. The Platform Pantilt measures $7.5 \times 7.5 \times 12$ cm and can move at rotational velocities up to 100^o/sec. The precision is somewhat limited, however, due to the use of stepper motors.

3.2 Background

One source of frustration is acquiring or building, then calibrating and controlling the mechanism itself. Until recently, no manufacturers have provided pan-tilt mechanisms specifically designed for computer control of cameras. Low-speed inexpensive motorized pan-tilt camera platforms are now available, for example from Edmund Scientific (catalog number F38,485), but these have no computer controls. Remote control pan-tilt devices intended for security camera applications, for example the Graystone Model V370PT, tend to be too slow (6 degrees per second) and too heavy (16 lbs.) for many robotic applications. Few of these devices have computer controls or position feedback information. The motion picture industry has also developed computer controlled pan-tilt devices. One example is the Kaleidoscope

Hothead II, made by Shepperton Film Studios. Although a digital interface is available, this device is intended to carry a large load such as a 35mm motion picture camera, and although it achieves relatively high speed (140^o/sec), it is very expensive. Another source of pan-tilt mechanisms is the lighting industry, which applies them to stage lighting, exhibitions and advertising. A high-speed computer controlled pan-tilt mechanism for stage lights is available from Multiline, but its bracketing and high cost make it unappealing for robot vision. A number of manufacturers (for example Aerotech, Daedel, Unislide, and Klinger Scientific) make computer controlled mechanisms that can be assembled into pan-tilt devices. Better suited for manufacturing and optics applications, the high resolution of these devices (less than 1 arc minute) makes them overkill for many computer vision applications and contributes to their high cost.

Perhaps because of their familiarity and availability, if not their generality and programmability, robot arms have often been the choice of vision researchers trying to actuate their cameras. Baloch and Waxman used a 5-axis robot arm mounted on top of a mobile robot as a camera pointing mechanism [8]. Allen also reported mounting a TV camera on a robot arm [4]. Raviv used a cartesian manipulator to implement camera pan, tilt, roll and translation [40].

Finally, a number of researchers including ourselves have embarked on building their own pan-tilt devices from scratch. Krotkov built what is now recognized as the first robot head, a computer controlled mechanism for moving two cameras [33]. Abbot and Ahuja report an 11 degree of freedom mechanism to control pan, tilt, vergence, horizontal translation, and lens parameters [2,3]. From Osaka University, Kawarabayashi et. al. report building an active vision "head" to control pan, tilt, vergence, zoom and focus parameters of a pair of cameras [29]. Dickmanns also reports a "fast" two-axis pan-tilt device carrying two cameras, one with a wide-angle view and the other telephoto, mounted atop their robotic automobile [20]. At Harvard, Clark and Ferrier constructed a seven degree-of-freedom "head" to control pan, tilt, and the vergence, focus and aperture of two cameras [16]. At least one two-eye system, the Rochester Robot, contains independent pan controls for two cameras on a tilting platform, in contrast to other systems in which vergence is coupled [13]. The pan-tilt mechanism in all of these designs is a MOM design.

Although many vision researchers are also interested in controlling parameters other than pan and tilt, we have restricted our attention in this thesis to these two degrees of freedom. We only mention here that pan-tilt mechanisms are not the whole story. We would like to build inexpensive computer controls for such camera parameters as focus, zoom, and aperture. In some cases it may be desirable to control camera roll and XYZ translation as well. With two or more cameras present, as in a stereo vision system, we would like to control the vergence angle between the camera as well as their baseline separation. These interesting subjects will remain outside the scope of the present discussion.

In this chapter we motivate the design and analyze the forward and inverse kinematics of the Platform Pantilt in order to prove that the linkage is mechanically sound.

3.3 Mechanism

The Platform Pantilt consists of a moving platform atached to a base with one fixed shaft and two shafts driven by linear stepper motors (Figure 3.1). The mechanism linking the stepper motors to the platform needs to transmit force and allow rotation while maintaining a 1-1 correspondence between motor and platform positions. This is accomplished by a combination of four universal joints and one pin joint.

Our original design had only one universal joint on each shaft. This, however, does not work because as the platform is rotated, the horizontal component of the distance between the joints decreases. The two-dimensional case depicted in Figure 3.3 shows what goes wrong with only one joint on the drive shaft. The shafts are rigidly fixed perpendicular to the base and to the platform. The distance between the joints is fixed. When the drive shaft extends, the angle between the drive shaft and the base must decrease which it cannot do.

One technique to overcome this problem, as is done with a Stewart Platform [22,24,52,53], is to attach

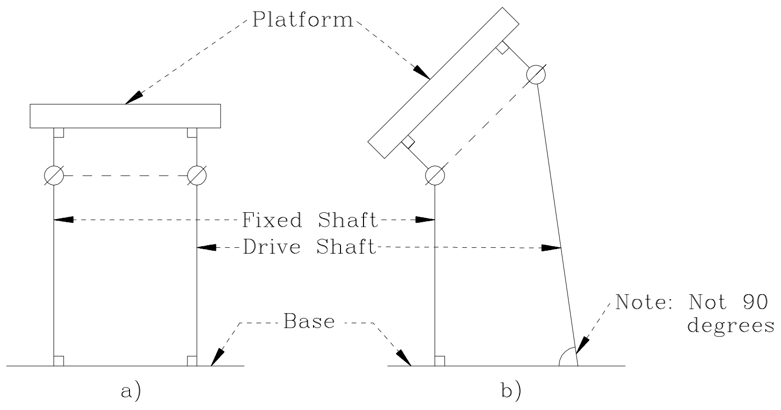


Figure 3.3: A non-working two-dimensional mechanism in two different positions with one pin joint on each shaft. Note that in b), the lower right angle is not fixed at 90^o as it should be.

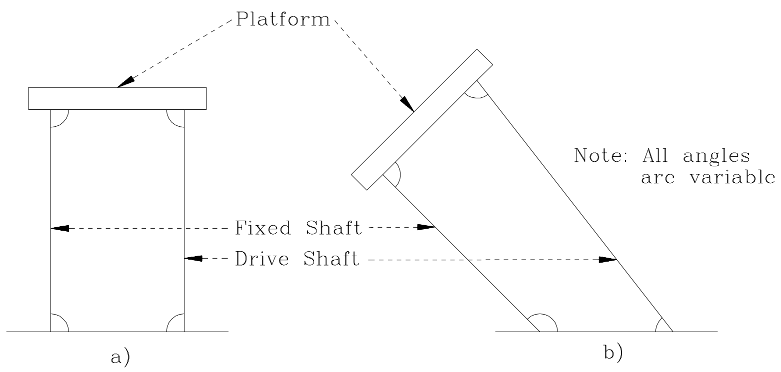


Figure 3.4: A two dimensional mechanism based on the Stewart Platform. Here, the drive shafts must be mounted on spherical joints.

the drive shafts to the base and to the platform with spherical joints (See Figure 3.4). Although this works in the two dimensional case, it does not work in three dimensions because the forward kinematics problem does not have a unique solution. The simplest description of the problem occurs when both drive shafts are fixed at the same height. In this configuration, the platform will have one uncontrollable degree of freedom as it rotates around the fixed shaft. This problem could be resolved by constraining the passive rotation of one of the motors to a single degree of freedom. However, this technique is not well-suited to miniaturization for two reasons: 1) It is difficult to build very small gimbal joints that could support the motor and platform; and 2) The motors themselves must be rotated which means that the motors must be large enough to counter their own rotational inertia.

The technique we have chosen avoids the problem of rotating the mass of the motors. The motors are fixed perpendicular to the base, and an extra joint is added to the drive shafts. Figure 3.5 illustrates the two dimensional case. Here, as the drive shaft extends and retracts (corresponding to h_1 increasing and decreasing in length), point C moves straight up and down. Point B is a fixed distance, s, from point C, and a fixed distance, d, from the fixed point A. So, point B moves along the point defined by the intersection of the two circles centered at A and C with radii d and s, respectively. Although there

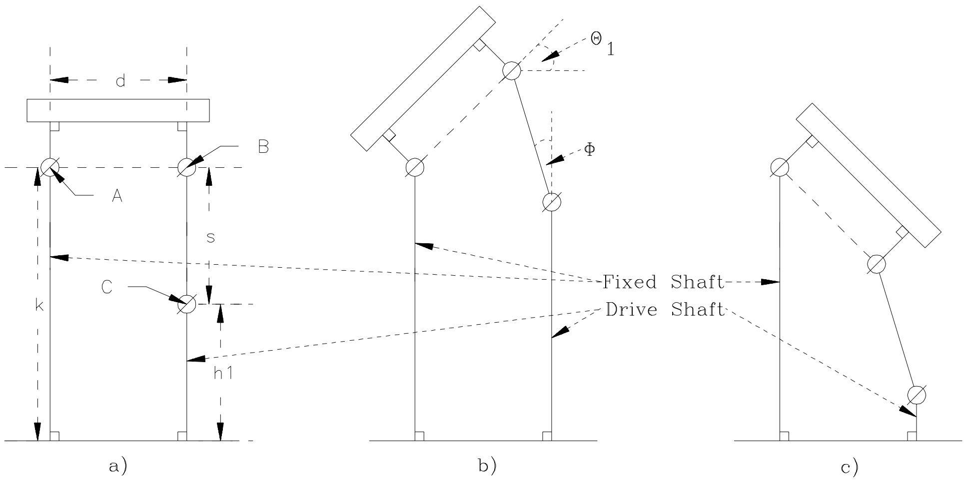


Figure 3.5: A two dimensional version of the mechanism we use shown in three positions. All the connections between the shafts and the platform and base are rigid. The dashed line between points A and B represents a fixed distance and does not represent a physical connection.

are two solutions for the point B, only one of them is physically attainable.

The straightforward extension of this mechanism to three dimensions by adding another drive shaft rigidly linked with two universal joints to the platform does not work. The platform would have one degree of uncontrollable freedom for all sets of drive shaft positions. The solution is to constrain one of the drive shaft linkages to two dimensions. This is easily done by replacing the lower universal joint on one of the drive shafts with a single degree of freedom pin joint. With this modification, the mechanism has no singularities. This is depicted in Figure 3.6.

With this approach, the kinematics are most easily analyzed by first looking at the two dimensional case and then extending the solution to three dimensions.

3.4 Forward kinematics in two dimensions

The forward kinematics problem for the two dimensional system (Figure 3.5) can be simply stated as "Given h_1, what is Θ_1?" That is, given the motor position, what is the angle of the platform. We define the origin to be at the bottom of the fixed shaft and we use homogeneous coordinates. We define the angle of the platform using the motor position and then solve using the known constraints. The angle of the platform is:

$$\Theta_1 = \sin^{-1}\left(\frac{B_y - A_y}{d}\right) \tag{22}$$

where A is the fixed point $(0, k, 1)$ and B is dependent on the motor position. C is the variable point $(d, h_1, 1)$. We can calculate B by taking the vector $(0, s, 1)$ rotated about an angle ϕ, and adding it to point C where ϕ is constrained by the fact that point B is a fixed distance d from point A.

$$B = (0, s, 1) \cdot [\text{rot } \phi] \cdot [\text{trans } C]$$

where

$$[\text{rot } \phi] = \begin{bmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

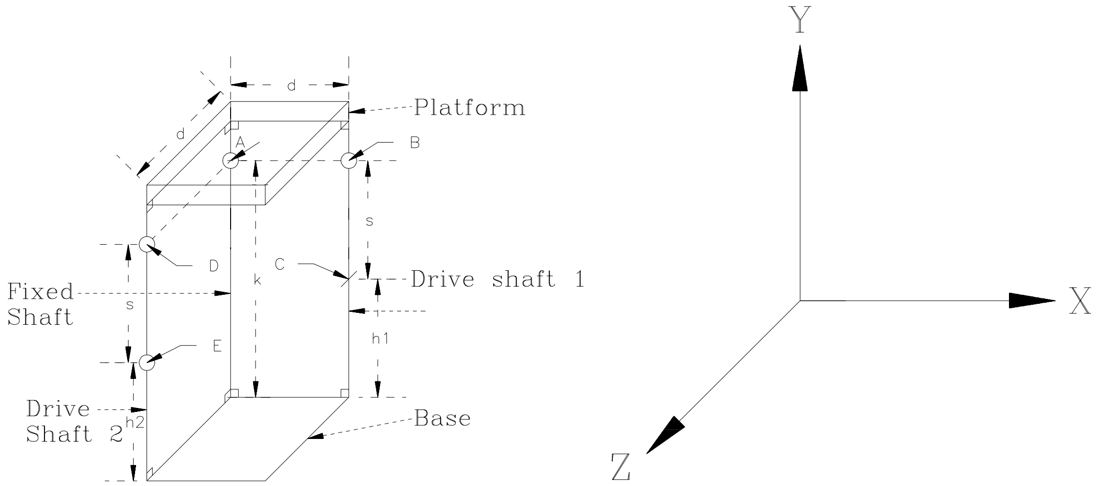


Figure 3.6: **left)** The three dimensional mechanism we use. Note that circles at points A, B, D and E denote universal joints with two degrees of freedom while the line at point C denotes a pin joint with only one degree of freedom. **right)** The coordinate system used in this section.

and

$$[\text{trans } C] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ C_x & C_y & 1 \end{bmatrix}.$$

Then, expanding B and substituting $d = C_x$ and $h_1 = C_y$, we can rewrite

$$B = (d - s\sin\phi,\ h_1 + s\cos\phi,\ 1). \tag{23}$$

Because B is a fixed distance d from point A, we add the constraint

$$(B_x - A_x)^2 + (B_y - A_y)^2 = d^2$$

and plug in the equations for B_x and B_y, using $A_x = 0$ and $A_y = k$ to get

$$(d - s\sin\phi)^2 + (h_1 + s\cos\phi - k)^2 = d^2. \tag{24}$$

We now want solve for ϕ. Expand Equation 24, plugging in $h_1 = C_y$, $k = A_y$ and $d = C_x$ to get

$$-2sC_x\sin\phi + s^2\sin^2\phi + (C_y - A_y)^2 + 2s(C_y - A_y)\cos\phi + s^2\cos^2\phi = 0$$

$$\underbrace{-2dC_x}_{a}\sin\phi + \underbrace{2s(C_y - A_y)}_{b}\cos\phi = \underbrace{-(C_y - A_y)^2 - s^2}_{c}$$

$$a\sin\phi + b\cos\phi = c \tag{25}$$

Using the triangle in Figure 3.7a, we can write

$$\sin\phi = \frac{m}{\sqrt{m^2+1}}$$

$$\cos\phi = \frac{1}{\sqrt{m^2+1}}$$

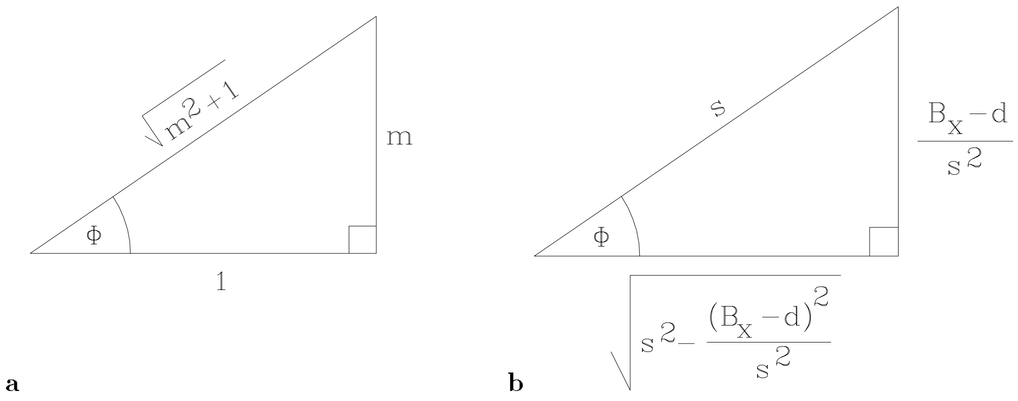


Figure 3.7: **a)** Triangle used to solve Equation 25. **b)** Triangle used to solve Equation 26.

Equation 25 becomes

$$\frac{am}{\sqrt{m^2+1}} + \frac{b}{\sqrt{m^2+1}} = c$$

$$\frac{(am+b)^2}{m^2+1} = c^2$$

$$a^2m^2 + 2abm + b^2 - c^2m^2 - c^2 = 0$$

$$(a^2 - c^2)m^2 + (2ab)m + (b^2 - c^2) = 0$$

$$m = \frac{-2ab \pm \sqrt{4a^2b^2 - 4(a^2-c^2)(b^2-c^2)}}{2(a^2-c^2)}$$

$$m = \frac{-2ab \pm 2c\sqrt{a^2+b^2-c^2}}{2(a^2-c^2)}$$

We can now solve directly for Θ_1 using Equation 22. From Equation 23, we have the value for B_y. We know that $A_y = k$, and $C_y = h_1$. Then,

$$\Theta_1 = \sin^{-1}\left(\frac{h_1 + s\cos\phi - k}{d}\right)$$

$$\Theta_1 = \sin^{-1}\left(\frac{h_1 - k + s/\sqrt{m^2+1}}{d}\right)$$

3.5 Inverse kinematics in two dimensions

The inverse kinematics problem for the two dimensional system can be stated by "Given Θ_1, what is h_1?" This is solved similarly to the forward kinematics problem. Now we know point B and we must find point C in order to find h_1.

$$h_1 = C_y$$

where

$$C = (0, -s, 1) \cdot [\text{rot } \phi] \cdot [\text{trans } B]$$

$$C = (B_x - s\sin\phi, B_y - s\cos\phi, 1)$$

and we know that $C_x = d$. We can expand C_x to get

$$B_x - s\sin\phi = d$$

We can write B as

$$B = (d, 0, 1) \cdot [\text{rot } \Theta_1] \cdot [\text{trans } A],$$

$$B = (A_x + d\cos\Theta_1, A_y + d\sin\Theta_1, 1)$$

We also know that $C_y = h_1$, so we can expand C_y, plugging in for B_y and get

$$h_1 = A_y + d\sin\Theta_1 - s\cos\phi \tag{26}$$

We know $\sin\phi$, but need $\cos\phi$. We can find it by using the triangle in Figure 3.7b. Then we can see that

$$\cos\phi = \frac{\sqrt{s^2 - (B_x - d)^2/s^2}}{s}$$

Plug this in to Equation 26 and get

$$h_1 = A_y + d\sin\Theta_1 - \sqrt{s^2 - \frac{(B_x - d)^2}{s^2}}$$

Plug in for B_x and get

$$h_1 = A_y + d\sin\Theta_1 - \frac{1}{s}\sqrt{s^4 - (a_x + d\cos\Theta_1 - d)^2}$$

3.6 Forward kinematics in three dimensions

The forward kinematics in three dimensions can be calculated in an analogous manner to the two dimensional case, but the math becomes extremely messy so an alternative approach is used. We assume the two dimensional case has already been solved and thus we know the position of point B. Then, given h_2, we want to know Θ_2.

$$\Theta_2 = \sin^{-1}\left(\frac{D_y - A_y}{d}\right) \tag{27}$$

Point A is known to be $(0, k, 0, 1)$, and point E is $(0, h_2, d, 1)$. Then, as point D is a fixed distance from points E, A, and B, it can be solved for by computing the intersection of the three spheres centered at those points. The equations describing these spheres are:

$$(D_x - E_x)^2 + (D_y - E_y)^2 + (D_z - E_z)^2 = s^2 \tag{28}$$

$$(D_x - A_x)^2 + (D_y - A_y)^2 + (D_z - A_z)^2 = d^2 \tag{29}$$

$$(D_x - B_x)^2 + (D_y - B_y)^2 + (D_z - B_z)^2 = 2d^2 \tag{30}$$

The solution for point D involves solving these three quadratic equations and three unknowns, D_x, D_y, and D_z, as follows. Expand these equations, rewriting the constants and we get:

$$D_x^2 + D_y^2 + D_z^2 + aD_x + bD_y + cD_z = d \tag{31}$$

$$D_x^2 + D_y^2 + D_z^2 + eD_x + fD_y + gD_z = h \tag{32}$$

$$D_x^2 + D_y^2 + D_z^2 + iD_x + jD_y + kD_z = l \tag{33}$$

where

$$\begin{aligned}
a &= -2E_x \\
b &= -2E_y \\
c &= -2E_z \\
d &= s^2 - E_x^2 - E_y^2 - E_z^2 \\
e &= -2A_x \\
f &= -2A_y \\
g &= -2A_z \\
h &= d^2 - A_x^2 - A_y^2 - A_z^2 \\
i &= -2B_x \\
j &= -2B_y \\
k &= -2B_z \\
l &= 2d^2 - B_x^2 - B_y^2 - B_z^2
\end{aligned}$$

Subtract Equation 32 from Equation 31 and Equation 33 from Equation 32 to get rid of the quadratic terms. We end up with equations for two planes.

$$(a-e)D_x + (b-f)D_y + (c-g)D_z = d-h \tag{34}$$

$$(e-i)D_x + (f-j)D_y + (g-k)D_z = h-l \tag{35}$$

The intersection of these two planes is a line, so we find its parametric equation and plug it into one of the original spheres (Equation 28). Note that because of the choice of the coordinate system, $a = e = g = k = 0$, and thus the coefficients of D_x in Equation 34 and of D_z in Equation 35 go to zero. Now pick two points, q_0 and q_1 on this line.

$$q_{0y} = 0 \;\Rightarrow\; q_{0z} = \frac{d-h}{c},\; q_{0x} = \frac{h-l}{-i}$$

$$q_{1x} = 0 \;\Rightarrow\; q_{1y} = \frac{h-l}{f-j},\; q_{1z} = \frac{d-h-(b-f)(h-l)/(f-j)}{c}$$

Note that these equations are not solvable when the denominator is equal to zero which occurs only at extreme positions. Now $q_0 + t(q_1 - q_0)$ on $t \in [-\infty, \infty]$ is the parametric formula for the line satisfying Equations 34 and 35. This can be plugged into Equation 28 to get one quadratic equation in t, which because $E_x = 0$, is equal to

$$[q_{0x} + t(q_{1x} - q_{0x})]^2 + [q_{0y} + t(q_{1y} - q_{0y}) - E_y]^2 + [q_{0z} + t(q_{1z} - q_{0z}) - E_z]^2 = s^2$$

Solve for t and plug into $q_0 + t(q_1 - q_0)$ to get point D, and finally plug into Equation 27 to get Θ_2.

3.7 Inverse kinematics in three dimensions

Fortunately, the inverse kinematics problem in three dimensions is much easier than the forward one. We want to find h_2 given Θ_2.

$$h_2 = E_y$$

We get E by taking the vector $(0,-s,0,1)$ rotated ψ_1 and ψ_2 about the x and z axes, respectively, and adding it to point D, where point D is calculated by taking the vector $(0,0,d,1)$ rotated Θ_1 and Θ_2 about the z and x axes, respectively, and adding it to point A.

$$\begin{aligned} D &= (0,0,d,1)\cdot[\text{rotx }\Theta_2]\cdot[\text{rotz }\Theta_1]\cdot[\text{trans }A] \\ &= (dsin\Theta_2 \sin\Theta_1, k - d\sin\Theta_2\cos\Theta_1, d\cos\Theta_2, 1) \end{aligned}$$

where

$$[\text{rotx }\Theta] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\Theta & \sin\Theta & 0 \\ 0 & -\sin\Theta & \cos\Theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ and } [\text{rotz }\Theta] = \begin{bmatrix} \cos\Theta & \sin\Theta & 0 & 0 \\ -\sin\Theta & \cos\Theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

$$\begin{aligned} E &= (0,-s,0,1)\cdot[\text{rotx }\psi_1]\cdot[\text{rotz }\psi_2]\cdot[\text{trans }D] && (36)\\ &= (D_x + s\cos\psi_1\sin\psi_2, D_y - s\cos\psi_1\cos\psi_2, D_z - s\sin\psi_1) && (37) \end{aligned}$$

Now we have the constraints $E_x = 0$ and $E_z = d$, so we get

$$E_z = d \;\Rightarrow\; \psi_1 = \sin^{-1}\left(\frac{d\cos\Theta_2 - d}{s}\right)$$

$$E_x = 0 \;\Rightarrow\; \psi_2 = \sin^{-1}\left(\frac{-d\sin\Theta_2\sin\Theta_1}{s\cos\psi_1}\right)$$

Finally, we can plug ψ_1 and ψ_2 into E_y in Equation 37

$$\begin{aligned} h_2 &= k - d\sin\Theta_2\cos\Theta_1 - \\ & s\cos\left(\sin^{-1}\left(\frac{d\cos\Theta_2 - d}{s}\right)\right)\cos\left(\sin^{-1}\left(\frac{-d\sin\Theta_2\sin\Theta_1}{\sqrt{s^2-(d\cos\Theta_2-d)^2}}\right)\right) \end{aligned}$$

3.8 Limits of motion

The limits of motion are depicted in Figure 3.8. The counterclockwise rotation is limited by the universal joints that can not rotate more that 90^o. To achieve this, length s must be greater or equal to length d. The clockwise rotation achieves its maximum when points A, B, and C are colinear. This occurs when $\cos\Theta = d/(d+s)$. So,

$$\Theta_{\max} = 90^o,$$

and

$$\Theta_{\min} = -\cos^{-1}\left(\frac{d}{d+s}\right).$$

3.9 Simulation of kinematics

To verify the correctness of the equations of this section, we implemented a software graphics simulation of both the forward and inverse kinematics. Figure 3.9 shows the solution of the forward kinematics problem with the Platform Pantilt shown in several positions.

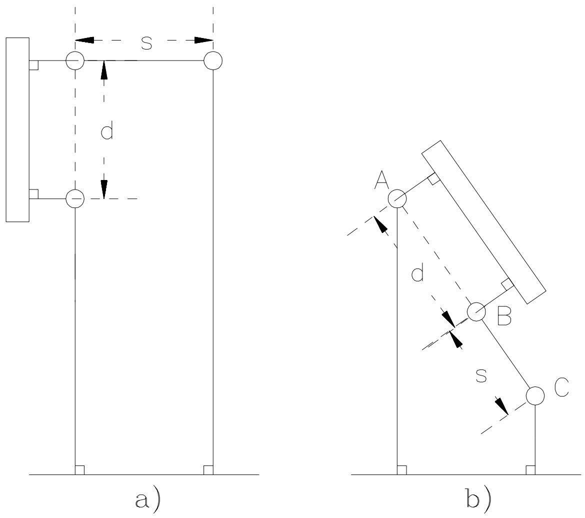


Figure 3.8: The limits of motion in two dimensions.

3.10 Prototype Platform Pantilt

The prototype Platform Pantilt was made with miniature 1″ diameter linear stepper motors. The linear motion has a maximum travel of 1.875″ and travels 0.004 inches/step. At a maximum of 550 steps per second, the motors can travel 2.2 linear inches/second. Our prototype motor has the following geometry (where the constants refer to Figure 3.6).

Parameter	Value
d	1.9 cm
s	2.8 cm
k	6.4 cm

The total size of the Platform Pantilt is $7.5 \times 7.5 \times 12$ cm. The platform has the maximum rotation, $\Theta_{max} = 90^o$ when $h_{max} = 6.1$cm and the minimum rotation, $\Theta_{min} = -66^o$ when $h_{min} = 2.0$cm. So the average theoretical maximum rotational velocity is equal to $(\Theta_{max} - \Theta_{min})/(h_{max} - h_{min}) \cdot 2.2\text{in/sec} = 214^o/\text{sec}$.

In practice, because stepper motors' strength is inversely proportional to their speed, they sometimes miss steps at high velocity. Thus the Platform Pantilt can be run reliably only at about 100 degrees/second.

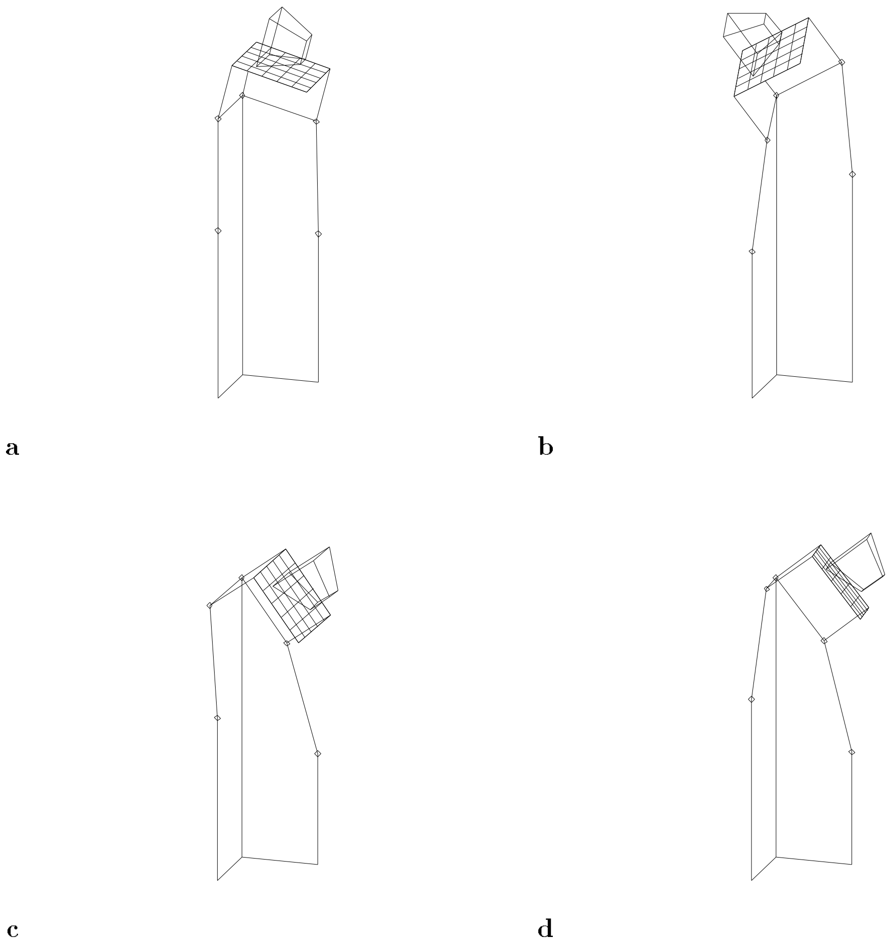


Figure 3.9: Simulation of the forward kinematics equations of this section. The Platform Pantilt is shown in four different positions.

Chapter 4

Spherical Pointing Motor

To overcome the limitations of the parallel motor approach, we designed and built the Spherical Pointing Motor[1] (SPM), a miniature single direct drive motor with two degrees of freedom. The SPM is capable of panning and tilting a load of 6 grams, for example, the CCD-based camera described in Chapter 2, at rotational velocities of up to several hundred degrees per second. Consisting of only wire wound around a metal form, a simple gimbal, and a small permanent magnet, the SPM is arguably the smallest and least expensive possible device capable of pointing a camera accurately and at high speed.

4.1 Spherical pointing motor theory

The Spherical Pointing Motor is an absolute positioning device, designed to orient a small camera sensor in two degrees of rotational freedom. The basic idea is to orient a permanent magnet to the magnetic field induced by three orthogonal coils by applying the appropriate ratio of currents to the coils. A simple way to understand this device follows: The net magnetic field of the three coils may be visualized as defining a vector (dipole) oriented at angles (Θ, Φ) on the unit sphere. The angles (Θ, Φ) are determined by the three coil currents. The rotor dipole then aligns itself with the net coil field to provide the actuation.

We have built two types of SPMs: one having the coils on the inside, free to rotate on a gimbal inside a fixed magnetic field, (Figure 4.2); the other having the coils on the outside and the magnets attached to the gimbal inside the coils (Figure 4.3). It is possible to build the smallest possible motor using the external coil design, so that is the approach we focus our attention on.

In this chapter, we look at the transfer function taking input current to pan and tilt angles. We point out some design constraints on the configuration of the coils and the permanent magnets, and show why the external coil design is preferable to the internal one. We discuss calibration of the motor, its dynamics and control.

Both designs are constrained by two principles that affect the range of motion of the motor. The SPM is meant to be used as a pointing device. As such, it has a *home position*, defined as the initial resting position from which the motor can make pan or tilt excursions of limited extent. Assuming that we want the home position to be centered within the possible excursions, we are led to the following constraints:

1. The permanent magnets must be positioned so that the field they define is orthogonal to both axes of rotation of the gimbal when it is in home position.
2. The camera must be positioned on the rotor so that its optical axis is orthogonal to both axes of rotation of the gimbal when it is in home position. Note that this is equivalent to being aligned

[1] Patent application #07/731,639, entitled "Spherical Pointing Motor" was submitted to the U.S. Patent office in 1991.

Figure 4.1: The Spherical Pointing Motor. At the center is a miniature camera consisting of a single CCD sensor chip and a lens assembly that fits on the rotor of this motor.

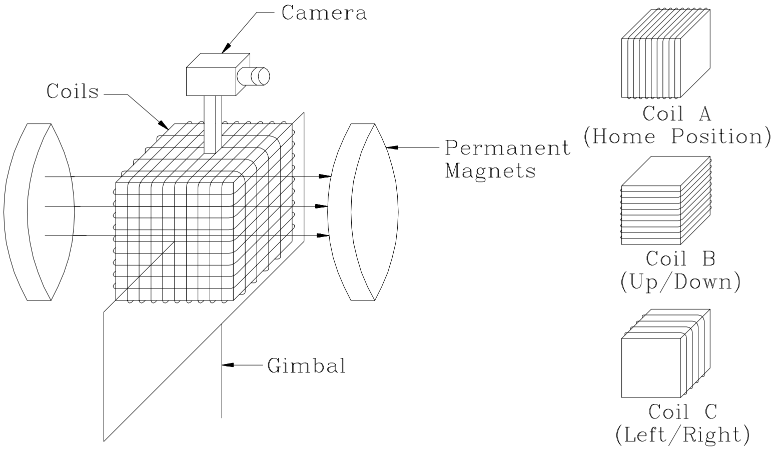


Figure 4.2: **left)** Illustration of the internal spherical pointing motor shown in its home position. **right)** Labels for the three coils.

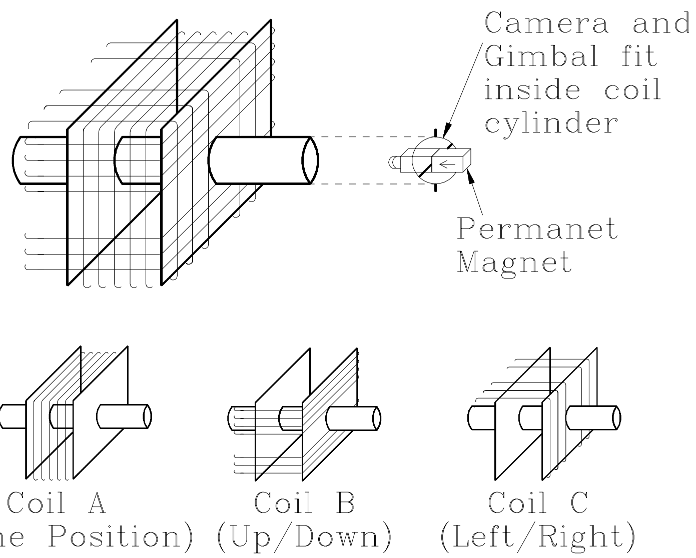


Figure 4.3: **top)** Illustration of the external spherical pointing motor shown in its home position. **bottom)** Labels for the three coils.

with the permanent magnetic field if the first design constraint is satisfied.

These design principles arise because the motor is limited to two mechanical degrees of freedom and because the motor rotation cannot be controlled about an axis aligned with the permanent magnetic field. To understand why this is so, we must examine the electromagnetic principle that provides the torque to move the motor.

The torque on the rotor comes from the basic electromagnetic principle that there is always an induced force on a current-carrying wire in a permanent magnetic field. Further, there will be a torque on a current-carrying wire loop in a permanent magnetic field in such a direction that the loop will move to make the normal to the plane of the loop align with the magnetic field. This is shown in Figure 4.4.

Given a coil of wire in a uniform magnetic field $\mathbf{B}$, the torque $\boldsymbol{\tau}$ exerted on the coil is the cross product

$$\boldsymbol{\tau} = \overline{\boldsymbol{\mu}} \times \mathbf{B} \tag{38}$$

where $\overline{\boldsymbol{\mu}}$ is the magnetic dipole moment having direction perpendicular to the plane of the coil and magnitude

$$|\overline{\boldsymbol{\mu}}| = N\boldsymbol{\iota}A \tag{39}$$

where N is the number of windings in the coil, $\boldsymbol{\iota}$ is the current in the wire, and A is the area enclosed by the coil [23,30]. The sign of $\overline{\boldsymbol{\mu}}$ is determined by the direction of current in the wire loop. When a nonzero current $\boldsymbol{\iota}$ flows through the loop, there will be a torque on the loop so that the loop will rotate to its minimum energy configuration where $\boldsymbol{\tau} = 0$, i.e. $\overline{\boldsymbol{\mu}}$ is aligned with $\mathbf{B}$. Note that in the SPM, the permanent magnetic field is not uniform around the area of the coil. This difference is discussed in Section 4.3.

Equation 38 implies that the torque is maximum when the angle between $\overline{\boldsymbol{\mu}}$ and $\mathbf{B}$, $\Psi = 90$ and drops to nothing when $\Psi = 0$. When the coil is at position $\Psi = 0$, it is at its minimum potential energy and there is no force on it. Because of this relationship between torque and angle, the friction of the bearings must be minimized. The precision of the motor is inversely related to the amount of friction in

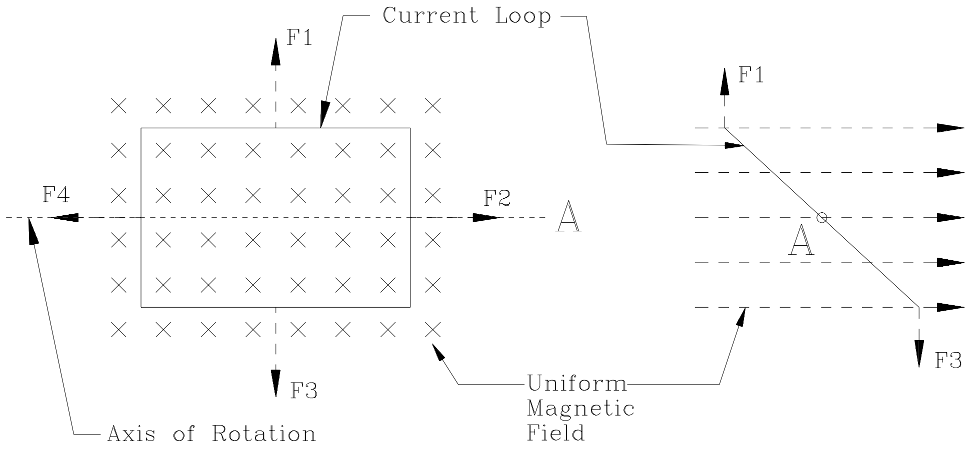


Figure 4.4: Torque on a current loop. **left)** Current loop rotates about horizontal axis with clockwise current. Uniform magnetic field normal to the page. Forces $\mathbf{F_2}$ and $\mathbf{F_4}$ are aligned with the axis and produce no net force. Forces $\mathbf{F_1}$ and $\mathbf{F_3}$ rotate the coil so that the coil encloses as many magnetic field lines as possible which occurs when the normal to the plane of the coil is aligned with the magnetic field. **right)** The same loop, but viewed from the side. (Adapted from [23, P. 541])

the bearings. This is because fine movements of the motor occur at very small angles where the torque is very small, and such fine movements must overcome the friction of the bearings.

If the first design constraint is not satisfied, then it is not possible to both pan and tilt the motor from the home position. Coils A and B control the tilting while coils A and C control the panning (Figures 4.2 and 4.3). When the motor has panned 90^0, coil B has kept its original orientation with respect to the magnets, but coils A and C have swapped their relative orientation. This is depicted in Figure 4.5 for the internal coil motor and the analog for the external coil motor is in Figure 4.6. In this position, the pan angle is controlled, but there is no way to control the tilt angle in this orientation (vice versa for the external coil design). This is because the axis of rotation is aligned with the magnetic field and no torque can be exerted around this axis. This is illustrated in Section 4.2.3. Without the first design constraint, the home position could be as depicted in Figure 4.5 in which case the motor could not be tilted along the vertical meridian.

The second design constraint stems from assuming that the two degrees of freedom desired for the camera are pan and tilt. If instead roll (rotation about the optical axis) and either pan or tilt is desired, then the second design constraint is not necessary. This constraint comes directly from the mechanical degrees of freedom available. If pan and tilt are desired, the camera's optical axis must be orthogonal to the two degrees of mechanical freedom.

The design constraints result in an undesirable effect for the internal coil motor. Because the first design constraint requires that the permanent magnets must also be positioned orthogonally to both axes of rotation, the optical axis of the camera must be aligned with the permanent magnetic field. Since the camera is between the two permanent magnets, its field of view will be obscured by the magnets. Therefore, some method of looking over the magnets must be found. This could possibly be done with mirrors, or as depicted in Figures 4.3 and 4.6, the camera could be mounted on a stalk. This gives the external coil motor a clear advantage because the camera is located in the center of the motor. This not only decreases the moment of inertia, but also allows the camera to be rotated around the focal point of the lens which is usually desired for machine vision applications.

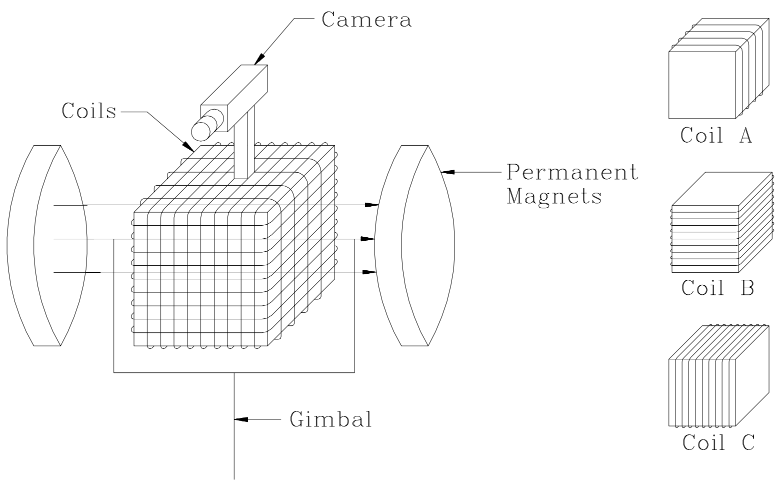


Figure 4.5: **left)** The internal coil motor panned 90^0 so that tilting is not possible. **right)** The coil labels for the motor in this position. Notice how coils A and C have swapped orientations.

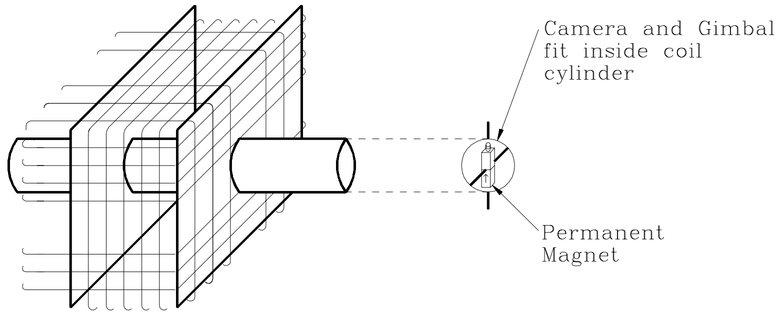


Figure 4.6: The external coil motor tilted 90^0 so that panning is not possible.

4.2 Theoretical Calculation of motor position

We can calculate the relation between motor orientation and coil currents by realizing that the current in each coil induces a magnetic field normal to the plane of the coil, and that the magnetic fields from the three coils add vectorially. Thus, the ratio of the coil currents determines the motor orientation. Applying a set of currents to the coils creates a torque on the rotor that will turn it to the position of lowest potential energy, so the SPM is an absolute positioning device. In this section, we assume ideal coils, i.e., they are perfectly symmetrical, have the same number of turns, and are the same size. We drop these assumptions in Section 4.3.

4.2.1 Position given currents

From Equation 38, we know that the torque on the rotor is proportional to the sine of the angle between the normal to the coil and the permanent magnetic field. If $\overline{\boldsymbol{\mu}}$ is aligned with $\mathbf{B}$, pointing in either the same or the opposite direction, there is no torque. However, when when they are pointing in the same direction, the potential energy is minimum and the coil is in a stable resting state. If they are pointing in opposite directions, the potential energy is maximum and the coil is in an unstable resting state. If the coil is slightly perturbed, it will swing around 180^0 to reach the minimum potential energy state.

We will use the external coil motor as an example, but the following derivation applies equally to the internal coil motor. We talk about torque on the magnets for the external coil motor, but an equal torque acts on the coils for the internal coil motor. For example, the torque on the magnets due to coil A for the external coil motor is equivalent to the torque on coil A for the internal coil motor.

We will use the coil labels shown in Figure 4.3. As the torque from each coil is dependent on the motor orientation, we must calculate when the sum of the torques due to all three coils vanishes. The destination position of the motor results from satisfying the equation

$$\boldsymbol{\tau}_A + \boldsymbol{\tau}_B + \boldsymbol{\tau}_C = 0 \tag{40}$$

where $\boldsymbol{\tau}_A$, $\boldsymbol{\tau}_B$, and $\boldsymbol{\tau}_C$ are the torques on the rotor due to coils A, B, and C. We define the coordinate system we will use in Figure 4.7. Here, both the permanent magnetic field and the home position are defined to lie in the direction of the positive x-axis. Θ (or tilt angle) is defined as the positive rotation about the z-axis and Φ (or pan angle) is defined as the negative rotation about the y-axis.

Let us examine the torque due to coil A. In home position, the magnetic field, $\mathbf{B} = (1,\ 0,\ 0)$, and coil A has the magnetic dipole moment, $\overline{\boldsymbol{\mu}}_A = N_A A_A \boldsymbol{\iota}_A(1,\ 0,\ 0)$. We use here, the constant $K_A = N_A A_A$. To calculate the torque from coil A at position $(\Theta,\ \Phi)$, we rotate $\overline{\boldsymbol{\mu}}_A$ by Θ around the z-axis and then by Φ around the y-axis, yielding the new dipole moment, $\overline{\boldsymbol{\mu}}'$.

$$\begin{aligned} \overline{\boldsymbol{\mu}}'_A &= K_A \boldsymbol{\iota}_A (1,\ 0,\ 0) \cdot [\mathrm{rot}_Z(\Theta)] \cdot [\mathrm{rot}_Y(\Phi)] \\ &= K_A \boldsymbol{\iota}_A (\cos\Theta \cos\Phi,\ \sin\Theta,\ -\cos\Theta \sin\Phi) \end{aligned}$$

Then,

$$\begin{aligned} \boldsymbol{\tau}_A &= \overline{\boldsymbol{\mu}}'_A \times \mathbf{B} \\ &= K_A \boldsymbol{\iota}_A (0,\ -\cos\Theta \sin\Phi,\ -\sin\Theta) \end{aligned}$$

We can calculate the torque from coils B and C similarly.

$$\begin{aligned} \overline{\boldsymbol{\mu}}'_B &= K_B \boldsymbol{\iota}_B (0,\ 1,\ 0) \cdot [\mathrm{rot}_Z(\Theta)] \cdot [\mathrm{rot}_Y(\Phi)] \\ &= K_B \boldsymbol{\iota}_B (-\sin\Theta \cos\Phi,\ \cos\Theta,\ \sin\Theta \sin\Phi) \\ \boldsymbol{\tau}_B &= \overline{\boldsymbol{\mu}}'_B \times \mathbf{B} \\ &= K_B \boldsymbol{\iota}_B (0,\ \sin\Theta \sin\Phi,\ -\cos\Theta) \end{aligned}$$

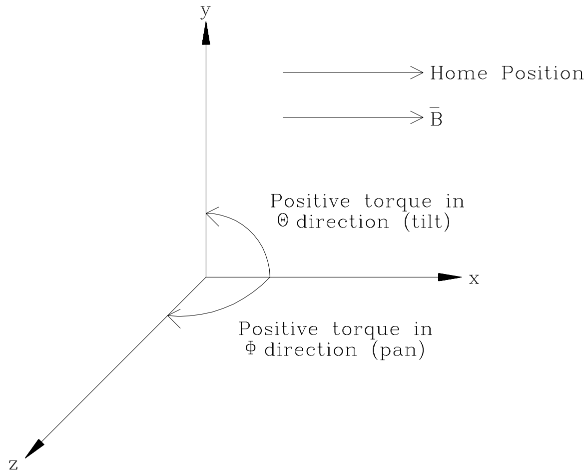


Figure 4.7: The coordinate system used for calculations. The permanent magnetic field and the home position both lie on the direction of the positive x-axis.

and

$$\begin{aligned}
\overline{\boldsymbol{\mu}}'_C &= K_C\boldsymbol{\iota}_C(0,\ 0,\ 1)\cdot[\mathrm{rot}_Z(\Theta)]\cdot[\mathrm{rot}_Y(\Phi)] \\
&= K_C\boldsymbol{\iota}_C(\sin\Phi,\ 0,\ \cos\Phi) \\
\boldsymbol{\tau}_C &= \overline{\boldsymbol{\mu}}'_C \times \mathbf{B} \\
&= K_C\boldsymbol{\iota}_C(0,\ \cos\Phi,\ 0)
\end{aligned}$$

Finally, we calculate the position using Equation 40.

$$K_A\boldsymbol{\iota}_A(0,\ -\cos\Theta\sin\Phi,\ -\sin\Theta) + K_B\boldsymbol{\iota}_B(0,\ \sin\Theta\sin\Phi,\ -\cos\Theta) + K_C\boldsymbol{\iota}_C(0,\ \cos\Phi,\ 0) = 0$$

This results in the solution

$$\Theta = \tan^{-1}\left(\frac{-K_B\boldsymbol{\iota}_B}{K_A\boldsymbol{\iota}_A}\right) \quad (41)$$

$$\Phi = \tan^{-1}\left(\frac{K_C\boldsymbol{\iota}_C}{K_B\boldsymbol{\iota}_B\sin\Theta - K_A\boldsymbol{\iota}_A\cos\Theta}\right) \quad (42)$$

Thus, the position of the motor (Θ, Φ) can be calculated given the currents applied to the three coils.

4.2.2 Currents given position

Given a desired motor position, we would like to calculate the coil currents necessary to attain that position. As previously stated, because there are three coils and only two mechanical degrees of freedom, the motor position depends upon the ratio of the currents in the coils. Thus, we can fix one of the coil currents, and calculate the other relative currents. Then the scale of the three currents will control the torque with which the motor is moved to the specified position. Note that the three degrees of freedom corresponding to the three coil currents are (Θ, Φ), and magnitude of the torque, $|\boldsymbol{\tau}|$.

Assume ι_A constant. Then from Equations 41 and 42,

$$\iota_B = \frac{-K_A}{K_B}\iota_A \tan\Theta \tag{43}$$

$$\iota_C = \frac{-K_A\iota_A}{K_C}(\sin\Theta\tan\Theta + \cos\Theta)\tan\Phi \tag{44}$$

The three currents are then scaled so that the largest current magnitude is set to a maximum, ι_{max}. That is,

$$\max(|\iota_A|, |\iota_B|, |\iota_C|) = \iota_{max}$$

Figure 4.9 shows plots of the calculated currents vs. position for each of the three coils along with the actual currents produced from the calibration procedure described in Section 4.3.

4.2.3 Illustration of design constraint

Using these definitions for Θ and Φ, we can now explain the first design constraint more thoroughly which is that the permanent magnet must be positioned so that the field it defines is orthogonal to both axes of rotation of the gimbal when it is in home position. As previously stated, this constraint comes from the fact that when the motor is panned 90^o (i.e., $\Phi = 90^o$), there is no control of the tilt (i.e., Θ is undefined).

Here we show that setting $\Phi = 90^o$ results in Θ being undefined. The case where K_A or K_B or K_C equals 0 represents a motor with one or more coils missing. This is a trivial case, so we assume that $K_A, K_B, K_C > 0$. Now plug $\Phi = 90^o$ into Equation 42, and we get

$$\tan^{-1}\left(\frac{K_C\iota_C}{K_B\iota_B\sin\Theta - K_A\iota_A\cos\Theta}\right) = 90^o$$

$$\frac{K_C\iota_C}{K_B\iota_B\sin\Theta - K_A\iota_A\cos\Theta} = \infty$$

This implies that $\iota_c \neq 0$ and that

$$K_B\iota_B\sin\Theta - K_A\iota_A\cos\Theta = 0$$

$$\tan\Theta = \frac{K_A\iota_A}{K_B\iota_B}$$

Equate this to the definition of $\tan\Theta$ in Equation 41 to get

$$\frac{K_A\iota_A}{K_B\iota_B} = \frac{-K_B\iota_B}{K_A\iota_A}$$

$$(K_A\iota_A)^2 = -(K_B\iota_B)^2$$

This is satisfied only when $\iota_A = \iota_B = 0$. But by Equation 41, $\tan\Theta$ is defined by $(-K_B\iota_B)/(K_A\iota_A)$ which is undefined when $\iota_A = \iota_B = 0$. Therefore, when $\Phi = 90^o$, Θ is undefined.

4.3 Calibration

The calculation of motor position for specified currents would be accurate in the ideal case where all three coils were perfectly symmetrical, had the same number of turns, and were the same size. In practice, none of these assumptions could be satisfied. Thus, the calculated currents give only an approximation to the actual position of the motor.

The motor must be calibrated to associate motor positions with the related set of currents that moves the motor to these positions. Only a small number of calibration points are necessary (depending on

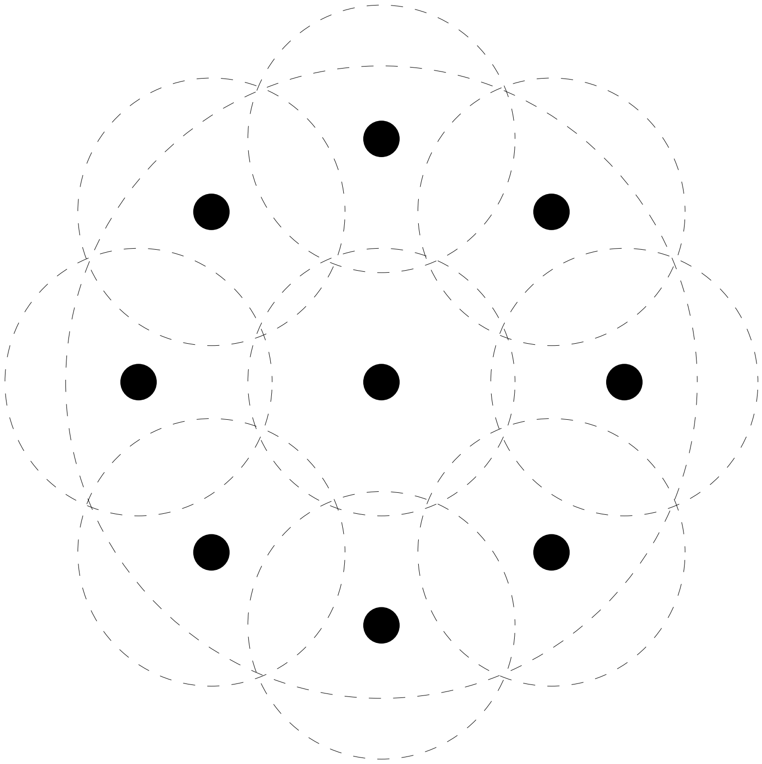


Figure 4.8: The solid dots comprise the scene used to calibrate the SPM. The nine smaller dashed circles are not part of the calibration pattern, but represent the field of view of the camera when centered on each dot. The single large dashed circle is also not in the scene, but represents the workspace of the motor. The entire workspace of the motor is covered by the small dashed circles.

the desired precision of the motor) since points between the calibrated positions can be approximated with linear interpolation.

We developed a procedure for automatic calibration of the SPM. It is based on image feedback from a camera mounted on the rotor of the motor. It assumes that a calibrated image sensor and lens are used, i.e., that it is known how many degrees each pixel subtends, and that this is constant throughout the field-of-view. The calibration algorithm uses a scene of black dots on a white backgound. For each motor position that is to be calibrated, the algorithm moves the motor approximately to that position using the calculated currents of Section 4.2. The algorithm analyzes the image and uses the position of the relevant dot to calculate the actual position of the motor. It then associates this position with the coil currents and stores it in a look-up-table, `calib`.

The workspace of the motor is larger than the field of view of the camera, so the scene must consist of more than one dot. To make the calibration procedure as automatic and as easy to use as possible, we do not want to measure the scene externally. Instead, the algorithm should work with only approximate knowledge of the positions of the dots. The only requirement is that there are enough dots so that the entire workspace of the motor is covered, and that it is possible to view the dots in pairs so that the position of one dot can be determined by its position relative to that of a previously determined one. In this way, the position of each dot can be precisely measured automatically by the calibration procedure. The scene used to calibrate the prototype motor is shown in Figure 4.8.

The algorithm, **calibrate**, calibrates the SPM with a multiple dot scene and it uses the the algorithms

center_dot, **find_dot** and **centroid**. It assumes the scene is arranged so that the center dot is initially approximately centered in the motor workspace, and thus in the camera field of view when the motor is in home position. A data structure is used supplying information about the arrangement of the scene. The algorithms are specified for space-variant images as defined in Chapter 1. The constants are defined in Appendix C. For each dot in the calibration pattern, the data structure contains:

dot_scene table

Variable	Description
motor_init	(Θ, Φ) position of where to initially move the motor for this point. This position should be defined so that the new dot and a previously calibrated dot are both in the field of the view of the camera when the motor is moved to this position.
new_coord	Estimated position of new dot in scene at motor position, motor_init
old_coord	Estimated position of previously calibrated dot in scene at motor position, motor_init
old_id	Index into this data structure to identify which previously calibrated dot is used

In these algorithms, the dot is defined as a single connected component whose values are below a threshold that is assumed to have been previously computed. Three routines are called that are not explicitly defined here. **soap_bubble** uses a standard relaxation technique to fill in the undefined values of the calibration table, calib. **median_filter** applies a standard 5×5 median filter to calib to eliminate bad points. **smooth** applies a standard 3×3 smoothing filter to calib to smooth noise.

The prototype SPM has been calibrated using **calibrate** and the scene depicted in Figure 4.8. The results are shown in Figure 4.9 along with the ideal case where the currents for each coil are shown over a workspace of $\pm 30^o$ in Θ and Φ.

Algorithm: **calibrate**

Function: This calibrates the SPM, creating a look-up-table associating coil currents with specific motor positions. This is accomplished automatically, imaging a scene with multiple dots. For each position to be calibrated, the motor is moved to that position using the calculated values. The dot is then searched for starting at its estimated position. Its actual position is determined and then associated with the motor position and entered in the calibration table.

Input: dot_scene table

Output: Calibration table: calib[Θ, Φ] holds currents for each coil at each calibrated motor position.

```
For each calibration point, [k,l], in calib
    calib[k,l] ← undefined
End For
Move motor to (0, 0), i.e., home position
For each dot in the scene, dot_index
    Move motor to motor_init[dot_index]
    Acquire image
                            { Find absolute position of current dot }
    zero_pt[dot_index] ← find_dot(new_coord[dot_index])
    If not the first dot Then
        old_dot_position ← find_dot(old_coord[dot_index])
        zero_pt[dot_index] ← zero_pt[dot_index] - old_dot_position +
            zero_pt[old_id[dot_index]]
    End If
    (CenterΘ, CenterΦ) ← center_dot(new_coord[dot_index])
    For each motor position, (Θ, Φ), to be calibrated with the new dot
        Move motor to estimated position, (Θ, Φ), using calculated table
        Acquire image
                            { Convert motor position to image position }
        estimated_dot_position ← (NROWS/2 - (CenterΘ - Θ)/DEG_PER_PIXEL,
            NCOLS/2 - (CenterΦ - Φ)/DEG_PER_PIXEL)
        new_dot_position ← find_dot(estimated_dot_position)
                            { Convert new_dot_position to calib indices }
        [k,l] ← DIM_CALIB/2 + (new_dot_position - zero_pt[dot_index]) ×
            DEG_PER_PIXEL × DIM_CALIB / DEG_OF_WORKSPACE
        If calib[k,l] = undefined Then
            calib[k,l] ← -1 × (Motor currents for each coil at this position)
        Else
            calib[k,l] ← (calib[k,l] - (Motor currents for each coil
                            at this position))/2
        End If
    End For
End For
calib ← soap_bubble(calib)
calib ← median_filter(calib)
calib ← smooth(calib)
```

Algorithm: **center_dot**

Function: This moves the motor to center the dot in the image. It is an iterative process, moving the motor to center the dot using the approximate values of the calculated motor position look up table.

Input: Starting motor position, (Θ, Φ)
Estimated dot position, p_1, in logmap image, L

Output: Motor position that results in centered dot.

Do
  Move motor to (Θ, Φ)
  Acquire image
  (dot_row, dot_col) $\leftarrow$ find_dot(p_1)
  $\Theta \leftarrow \Theta +$ (dot_row - NROWS/2) $\times$ DEG_PER_PIXEL
  $\Phi \leftarrow \Phi +$ (dot_col - NCOLS/2) $\times$ DEG_PER_PIXEL
  $p_1 \leftarrow$ (NROWS/2, NCOLS/2)
While dot not centered
return(Θ, Φ)

Algorithm: **find_dot**
Function: This finds the dot in the image starting at the specified location, searching outward. When the dot is found, its centroid is computed and returned.
Input: Starting image location, p_1, in logmap image, L
Output: Centroid of found dot in TV image coordinates.

For each pixel $p_2 \in L$
  visited[p_2] $\leftarrow$ **FALSE**
End For
visited[p_1] $\leftarrow$ **TRUE**
front $\leftarrow$ 0
back $\leftarrow$ 0
While (($L(p_1) \geq$ threshold) **AND** (front $<$ NPIXELS) **AND** ($a(p_1) > 0$)) **Do**
  For each neighbor $p_2 \in \mathcal{N}(p_1)$
    If (visited[p_2] = **FALSE**) **Then**
      visited[p_2] $\leftarrow$ **TRUE**
      recurse[back] $\leftarrow p_2$
      back $\leftarrow$ back + 1
    End If
  End For
  $p_1 \leftarrow$ recurse[front]
  front $\leftarrow$ front + 1
End While
return(centroid(p_1))

Algorithm: **centroid**

Function: This computes the centroid of a dot in an image given a point in the dot. It assumes the dot is a single connected component of pixels below the threshold. It computes the centroid in the logmap image, but weights the pixels by their area in TV pixels to get the true centroid. It returns the result in TV image coordinates.

Input: Starting image location, p_1, in logmap image L

Output: Centroid of found dot in TV image coordinates.

If ($a(p_1) = 0$) **Then**
    print("Error: dot searched for at invalid point")
    return(error)
End If
For each point $p_2 \in L$
    visited[p_2] $\leftarrow$ **FALSE**
End For
visited[p_1] $\leftarrow$ **TRUE**
$p_3 \leftarrow 0$
front $\leftarrow$ 0
back $\leftarrow$ 0
total_weight $\leftarrow$ 0
$\mathcal{P}_a \leftarrow 0$ { TV image point }
$\mathcal{P}_{center} \leftarrow$ (NROWS/2, NCOLS/2)
Do
    For each neighbor $p_2 \in \mathcal{N}(p_1)$
        If ((visited[p_2] = **FALSE**) **AND** ($L(p_2) <$ threshold)) **Then**
            weight $\leftarrow L(p_2) \times a(p_2)$
            $\mathcal{P}_a \leftarrow \mathcal{P}_a$ + weight $\times\ \boldsymbol{\mu}(p_2)$ { Convert p_2 to TV coordinates }
            total_weight $\leftarrow$ total_weight + weight
            visited[p_2] $\leftarrow$ **TRUE**
            recurse[back] $\leftarrow p_2$
            back $\leftarrow$ back + 1
        End If
    End For
    $p_1 \leftarrow$ recurse[front]
    front $\leftarrow$ front + 1
While (front $\leq$ back)
If (total_weight > 0) **Then**
    $\mathcal{P}_b = \mathcal{P}_a$/total_weight - $\mathcal{P}_{center}$
Else
    $\mathcal{P}_b = (0, 0)$
End If
return($\mathcal{P}_b$)

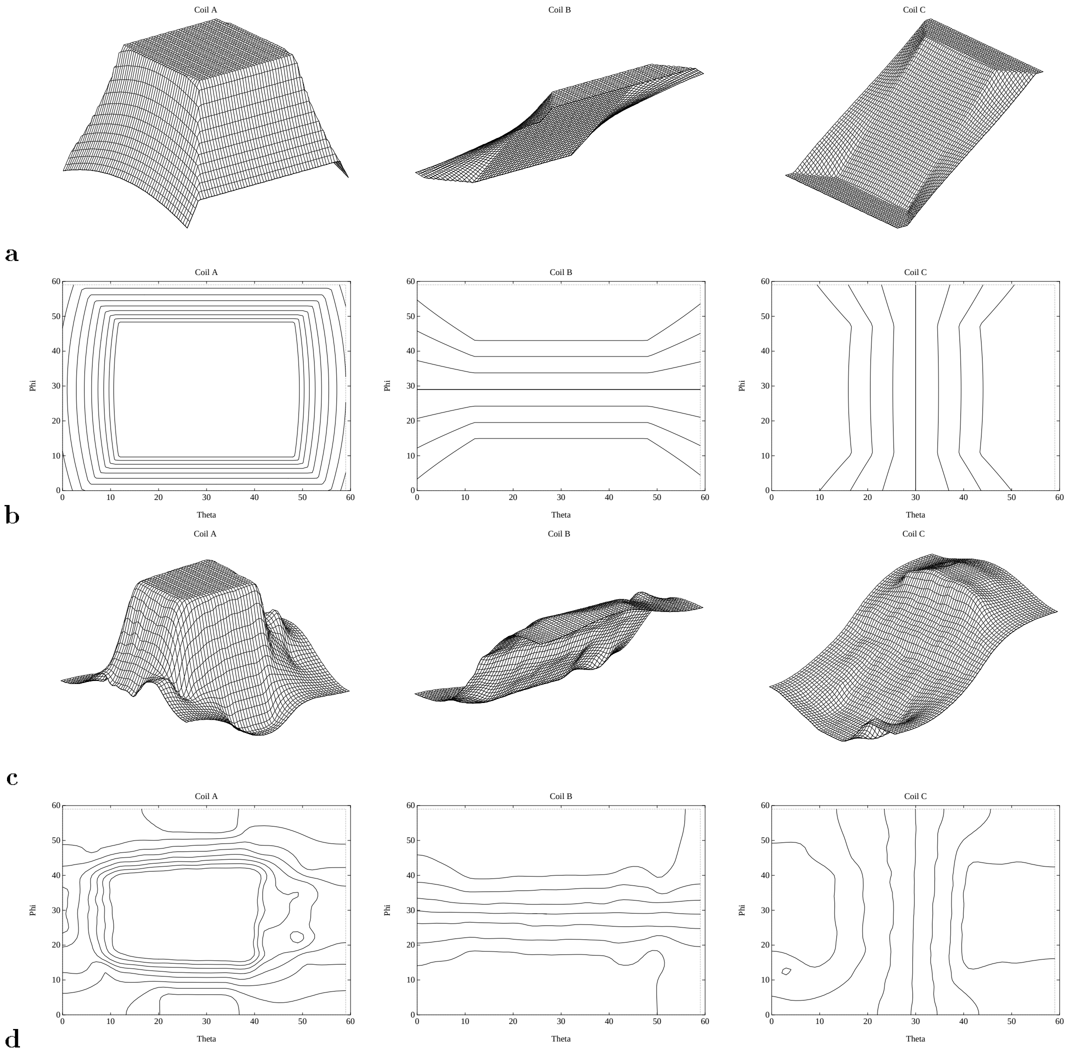


Figure 4.9: **a)** Theoretical calculated currents vs. position for each of the three coils shown as a mesh plot. **b)** The same as **a)** shown as a contour plot. **c)** Calibrated currents vs. position for each of the three coils shown as a mesh plot. **d)** The same as **c)** shown as a contour plot. The data shown here is for the $\pm 30^o$ workspace in (Θ, Φ).

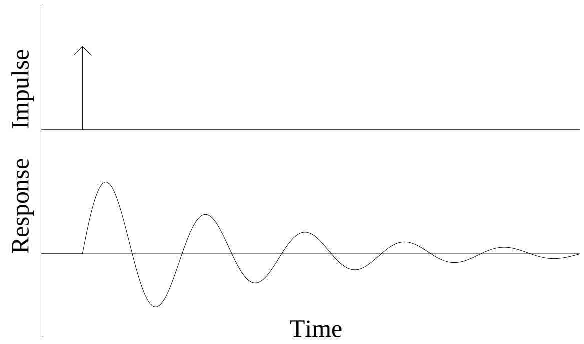


Figure 4.10: Model of system response to a single impulse.

4.4 Dynamics and control

The dynamics of the SPM are that of a simple second order system. When a new set of currents are applied to the coils, a torque is created that moves the rotor to the new position. We model this with a single impulse (Figure 4.10). We are interested in finding a way to reduce the oscillatory response without resorting to a closed-loop system.

We model the *ringing* of the SPM by using the fact (Section 4.2) that the torque on the rotor is proportional to the sine of the angle between the current position and the destination position. Let us call this angle Ψ. The motor will accelerate towards the destination position with the torque decreasing as it is reached. However, it will overshoot and ring around the final position. The torque on the rotor is $\boldsymbol{\tau} = \kappa \sin \Psi$, for some κ. If we use the approximation $\sin \Psi \approx \Psi$, then the position of the motor will follow Equation 45 [23, P. 228].

$$-\kappa\Psi - c\frac{\mathrm{d}\Psi}{\mathrm{dt}} = r\frac{\mathrm{d}^2\Psi}{\mathrm{dt}^2} \tag{45}$$

where c is the damping constant and r is the rotational inertia of the rotor. The approximate solution to this is

$$\Psi = Ce^{-ct/2r}\cos(\omega t + \beta) \tag{46}$$

where C and β are constants, and the exponential function describes the magnitude envelope of the ringing. The frequency of the ringing is $\omega \approx \sqrt{\kappa/r}$.

We measured the ringing of the SPM by recording the induced voltage of the ringing on a pickup coil of magnet wire that was placed adjacent to the SPM. The recording was made by amplifying the induced voltage and digitizing this through the audio port of a Sun Sparcstation at a sample rate of 8 KHz. Because the SPM is controlled with Pulse-Width Modulation (PWM), the pickup coil not only records the ringing (at approximately 60 Hz), but also records the PWM signal (at 1 KHz). The recorded signal is filtered with an FFT to isolate the ringing from the PWM in the induced signal. An example of the ringing shows the result of moving the SPM, uncontrolled, about 20^o (Figure 4.11).

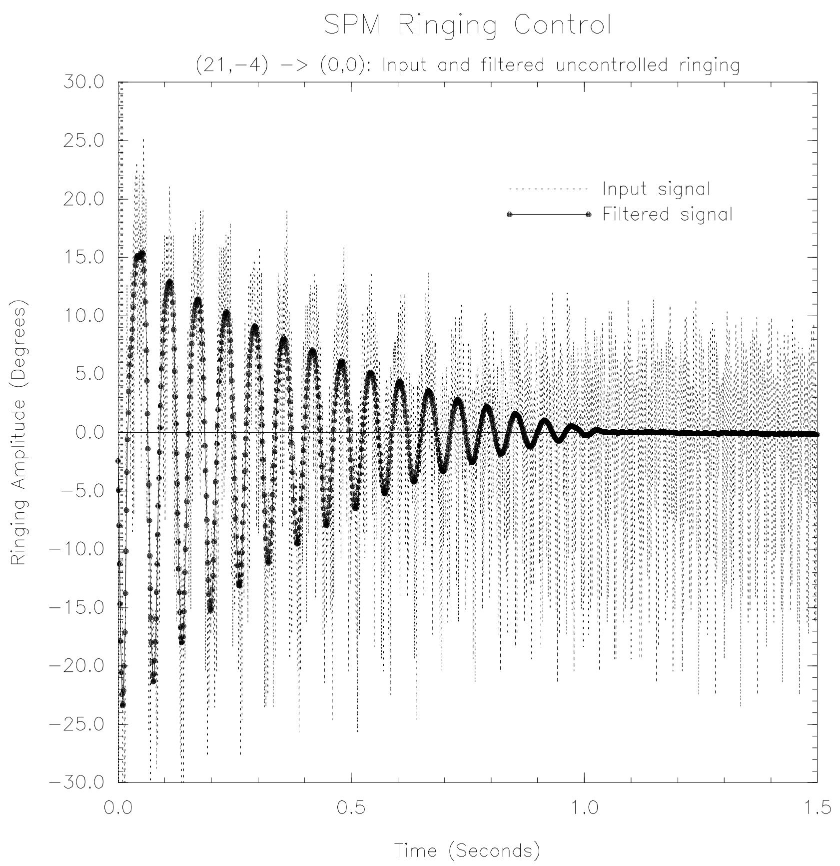


Figure 4.11: Uncontrolled ringing measurement taken by induced voltage on a separate pickup coil. The input signal (combination of ringing and PWM) is shown along with the filtered signal that isolates the ringing.

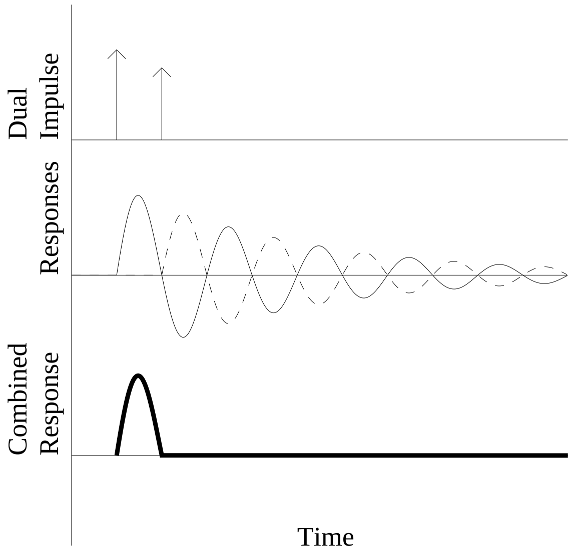


Figure 4.12: Ideal model of system response to a dual impulse.

This ringing can be greatly reduced by various open-loop control methods. Perhaps the simplest strategy is to move the motor from the initial position to the destination position in small decreasing increments at fixed time intervals. This way, maximum motor velocity is traded off for control. Including the time it takes for ringing to stop, the controlled approach yields a faster average velocity. This method is implemented by moving a fixed percentage of the distance between the current position and final position at each step. This will decrease the motor movement with each step. The motor velocity can then be controlled with either the percentage movement of each step, or the time interval between steps. We use a constant time interval, and vary the percentage movement of each step to experiment with different speeds. We call this the *fixed percentage* control method.

A second slightly more complicated open-loop control strategy based on traditional stepper motor control theory, and also examined by Singer and Seering [50], yields better results than the fixed percentage method. The idea, called the *two step* method, is based on the fact that the SPM rotor oscillates with an approximately constant period. We give two impulses (Figure 4.12) 180^o out of phase with each other so as to cancel the ringing. We implement this by first stepping the motor to a point midway between the initial and destination positions (determined from the calibration results of Section 4.3). We then wait for the point where the rotor velocity is zero at which point we apply a second step to hold the rotor at this position. The energy of the ringing will be largely dissipated, and the rotor will be at the destination position. Although we don't know exactly when to apply the second step, we can estimate it by waiting half the period of oscillation. Because we model the system with a one-dimensional pendulum when it actually is a two-dimensional problem, our estimation is not exact, but is a close approximation. In practice, because neither the midpoint nor the time at which the currents are changed are exactly correct, not all of the ringing energy is dissipated, and a little ringing of the same period remains, but with vastly reduced amplitude (Figure 4.13). This approach is unusual in that it takes a constant time to move any distance, where the time is half the period of oscillation. A sample result of this approach is shown in Figure 4.14.

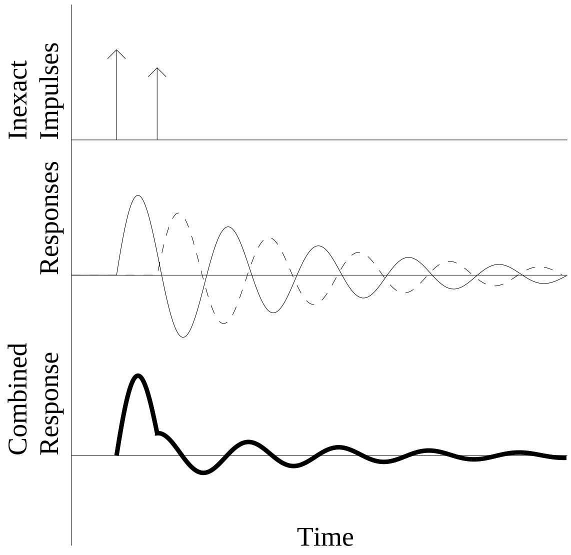


Figure 4.13: Model of system response with inexact timing to a dual impulse.

A problem with the two-step control strategy is that the system response is not very robust, i.e., a small uncertainty in the midpoint or delay timing results in relatively large uncontrolled ringing. We compensated for this imprecision by various combinations of the previous two control strategies. The first approach is to apply the two-step method twice, first moving to the midpoint, and then to the destination point. This method theoretically also takes constant time, which is twice as long as the two-step method. We call this approach the *dual two-step* method.

A final variation, called the *multi two-step* method, is to apply the two-step method multiple times with logarithmically decreasing steps. The motor is always moved half way between its current position and the destination position with the two-step method. This takes time $O(\log(\Psi))$ where Ψ is the total angular distance travelled.

We are currently investigating other control approaches advocated by Singer and Seering involving three impulses to make a more robust strategy. Results from the open-loop control strategies we tested are presented in Section 4.7.2.

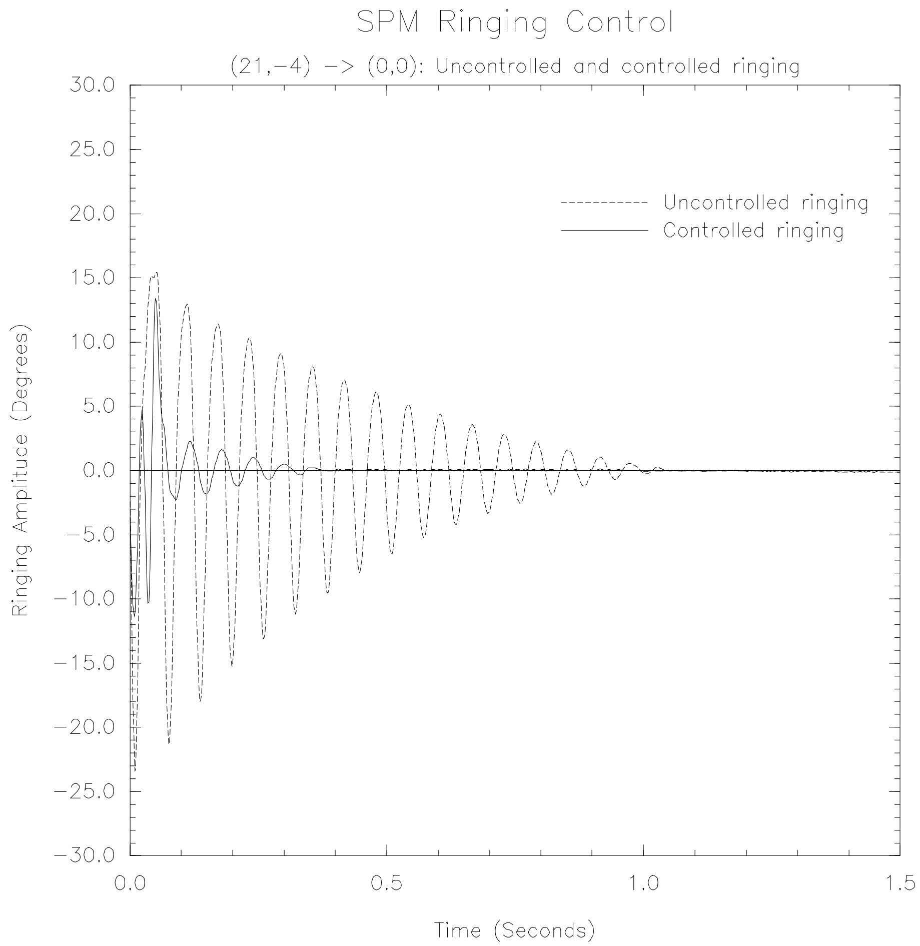


Figure 4.14: Measurement of two-step open-loop control strategy to reduce SPM ringing. Results are shown for the same movement of 21.4^o with and without control. The motor currents are changed to hold the motor at the final position when the motor is expected to have zero velocity. Because the midpoint and timing are not exactly correct, the ringing is not completely eliminated.

One of the motivating factors behind the development of the SPM was to make a motor that could be run by an open-loop controller, and we have accomplished this task. However, if higher performance is needed, a velocity or position sensor could be added to run the motor with a closed-loop control strategy. This would allow the two-step control strategy to applied more accurately, as the zero-velocity point would be known exactly. Not only would this increase the speed of the motor, but it would also increase its accuracy. The motor's supply currents could be 90^o out of phase with the position, like in traditional DC motors. This would result in always driving the motor at its maximum torque and thus the torque would no longer be dependent on the difference between the current and destination position. No closed-loop control strategies have been implemented yet.

4.5 Design Issues

We want to design the SPM to maximize its torque while minimizing its size and power usage. There are several parameters that we have control over in this design. The diameter of the wire and the area, number of turns, and voltage across the coil as well as the magnetic return path are the parameters that most affect the design, and are the ones that we will discuss here.

The first issue is the magnetic return path. Our magnetic circuit consists of a permanent magnet that generates magnetic field lines with a geometry dependent on the type of material in its immediate vicinity. Magnetic circuits can be described analogously to electric circuits. With electric circuits, different materials have different resistances and the current will follow the path of least resistance. In magnetic circuits, different materials have different permeability and the magnetic field will follow the path of greatest permeability. Because there is only torque on a current loop if the magnetic field goes through the loop and returns outside the loop, it is critical that the geometry of the motor is designed so that the permanent magnetic field lines return outside the coil. Air has very low permeability while iron has very high permeability. By adding an iron return path outside the motor, more magnetic field lines will enclose the coil because the magnetic field lines will follow the path of highest permeability, resulting in more torque. A simulation of the magnetic circuit for the external coil SPM with and without an iron return path is shown in Figure 4.15.

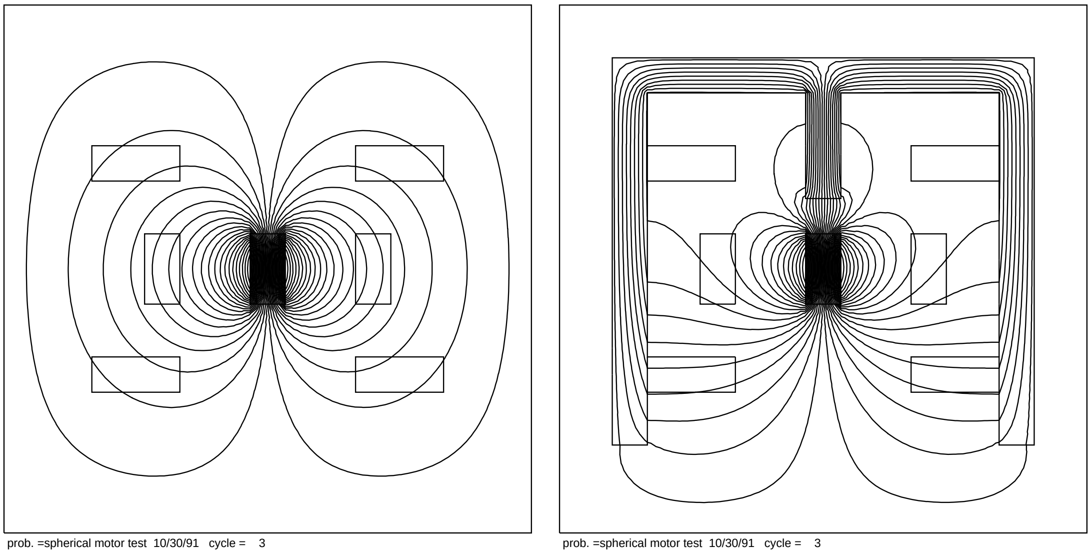


Figure 4.15: Simulation of magnetic circuit for the SPM **left)** without and **right)** with an iron magnetic return path. The permanent magnet is in the center. A camera mounted on the magnet would be looking towards the bottom of the page. The rectangles represent the coils of the motor. It can be seen that without the magnetic return, only a small percentage of the magnetic field returns outside the coils, but with the magnetic return, an efficiency of nearly 50% is achieved.

Now let us investigate the parameters of voltage ($\mathcal{V}$), the number of turns in the coil (N), the area of the coil (A), and the diameter of the wire (D). We know that $\boldsymbol{\tau} \propto N\boldsymbol{\iota}AB$, and power, $P = \mathcal{V}\boldsymbol{\iota}$. We want to maximize the ratio $\boldsymbol{\tau}/P$ while maintaining a minimum acceptable torque and maximum acceptable power. The resistance of the wire is proportional to the number of turns of the coil and the area of the coil, and it is inversely proportional to the square of the diameter of the wire, so we can write

$$\mathcal{R} \propto \frac{NA}{D^2}$$

Because $\boldsymbol{\iota} = \mathcal{V}/\mathcal{R}$, we can rewrite the equation for current as $\boldsymbol{\iota} \propto (\mathcal{V}D^2)/(NA)$. Finally we can rewrite the equations for torque, power and their ratio using only the input parameters:

$$\boldsymbol{\tau} \propto \mathcal{V}D^2 \cdot B \tag{47}$$

$$P \propto \mathcal{V}D^2 \cdot \frac{\mathcal{V}}{NA} \tag{48}$$

$$\frac{\boldsymbol{\tau}}{P} \propto \frac{BNA}{\mathcal{V}} \tag{49}$$

From these relationships, we can see that we want to maximize B and N while minimizing $\mathcal{V}$. In doing so, we must also maintain our torque and power requirements and adjust V, D, and A accordingly. The values of these parameters for our prototype SPM are reported in Section 4.7.

4.6 Controller description

The SPM is controlled by limiting the amount of current in each coil. This can be done either by regulating the voltage on the coils, by using a constant voltage with a Pulse-Width Modulation (PWM) technique, or by using a variable current source. The PWM method is preferred for ease of use, although the variable current source technique has the advantage that the currents, and thus the motor position, are independent of the temperature of the motor.

The SPM controller we use is based on a Motorola M68332 microcontroller that is capable of creating PWM signals. A Unitrode L293N power driver supplies the current for the motor. PWM means driving the motor with a square wave of constant amplitude and period with a variable duty cycle. The percentage of the period where the supply is high is called the duty cycle. A duty cycle of 0 gives no current to the motor while a duty cycle of 100 gives maximum current. The inertia and inductance of the coil effectively smooths the PWM supply. The polarity of the voltage on the coil also needs to be controlled in order to position the motor everywhere on the unit sphere. The power driver allows us to control the voltage polarity.

4.7 Prototype spherical pointing motor

The prototype SPM (Figure 4.1) is $4 \times 5 \times 6$ cm, weighs 160 grams and is capable of actuating a 6 gram load. Its total workspace is approximately 60^o in both the pan and tilt directions. The permanent magnetic field is estimated at 1 Tesla with an efficiency of about 5%. The number of turns in the coil is estimated. The maximum torque is computed here, where $\Psi = 90^o$. This table shows the parameters of the motor along with its calculated torque and power usage. The actual torque and power usage have not been measured.

Coil	Parameter	Value
Inside	N	630
	D	30 gauge (= .000254 m)
	A	0.000730 m^2
	$\mathcal{V}$	15 Volts
	R	21Ω
	τ	.016N $\cdot$ m or 2.3oz $\cdot$ in
	P	11 Watts
Middle	N	500
	D	30 gauge
	A	0.000993 m^2
	$\mathcal{V}$	15 Volts
	R	26.0Ω
	τ	.014N $\cdot$ m or 2.0oz $\cdot$ in
	P	9 Watts
Outside	N	420
	D	30 gauge
	A	0.00130 m^2
	$\mathcal{V}$	15 Volts
	R	22Ω
	τ	.019N $\cdot$ m or 2.6oz $\cdot$ in
	P	10 Watts

4.7.1 Accuracy and precision measurements

The precision and accuracy of the SPM were measured by reflecting a laser diode off a reflective surface attached to the rotor. The reflected beam's movement is measured on a wall several feet away as described below.

The precision of the motor is defined as the angular distance between adjacent positions. If the controller consisted of an analog variable current source, the motor's precision would be infinite. As we are using a digital PWM controller, the precision with which we can control the motor is limited by the step size of the PWM duty cycle. The step size is dependent on the frequency of the PWM. Lower frequencies have more precise control of the step. However, lower frequencies also introduce a choppiness that vibrates the motor. We use a 1KHz frequency and are able to specify 4,000 steps in the duty cycle (i.e., 4,000 different duty cycles). As the duty cycle of the PWM on a single coil with a specified polarity changes between 0 and 100, the motor will turn at most 45^o. Thus, we can position the motor to steps no smaller than $45^o/4000 = 0.011^o$ or 0.68 minutes of arc.

The accuracy of the motor is defined as the repeatability of the motor. If the motor were balanced so that gravity was not a factor, then its accuracy would be dependent only upon the friction of the bearings. As there is no iron core, there is no hysteresis, and thus it is absolutely repeatable. Because the motor, in fact, is not perfectly balanced, the motor position is not constant at different orientations to gravity, but at a fixed orientation, the accuracy is dependent only upon the friction of the bearings.

We measured the motor's accuracy with a laser setup (Figure 4.16) and found the motor to be accurate to 0.15^o. The camera mounted in our motor returns 192×165 pixels with a 33.6^o horizontal field of view. This is equivalent to 0.175^o per pixel. For our camera and lens, the SPM is thus accurate to about one pixel.

The procedure for measuring motor accuracy is as follows. The motor is set at a fixed position with the rotor positioned at the place to be measured. The laser is then oriented so that the reflected beam hits the wall at a right angle. This point on the wall is recorded. The rotor is moved away and then back to the original position. The distance between the new reflected laser position and the original position is measured, and the accuracy of the motor is calculated according to Equation 50.

The measurement setup is illustrated in Figure 4.16. The motor is moved Θ degrees. The distance from the motor to the wall is f, and e is the distance between the first and second laser position on the wall. Φ is the angular difference between the first and second motor positions. The solid lines represent the first motor position, and the dashed lines represent the second. The relation between the laser point movement and the accuracy of the motor is

$$\Phi = \frac{1}{2}\tan^{-1}(\frac{e}{f}) \tag{50}$$

4.7.2 Velocity measurements

We measured the average velocity of the motor for point to point motions since we have no ability to measure instantaneous velocity. We recorded the velocity using the technique described in Section 4.4 for measuring the ringing. The motor controller was programmed to accept motion commands with a specified control strategy over an RS-232 serial port. A program running on a Sun Sparcstation controlled the SPM and automatically recorded and analyzed the results. For each control strategy, ten measurements were made at each of ten different angular movements. The averaged results with error bars showing the variance of each experiment are shown in Figures 4.17, 4.18, and 4.19.

A summary of the control strategy measurements follows:

No control Base line for comparing control strategies.

Fixed percentage The motor is moved a fixed percentage of the distance between its current position and the destination position with a fixed delay between each movement.

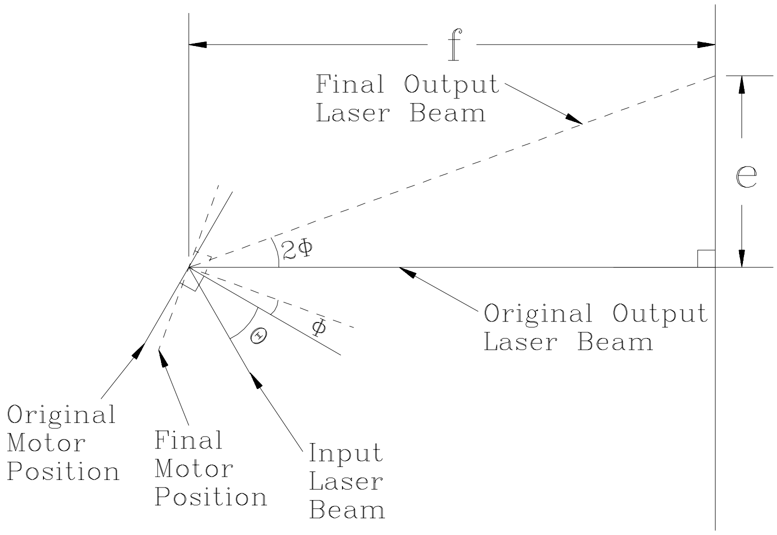


Figure 4.16: Illustration of our experimental setup for measuring motor accuracy. The laser beam is reflected off the rotor onto a flat surface. The motor is then moved to a new position and back to the original position. The new laser beam position is compared with the original position to see how accurately the motor returned to the original position.

By making the percentage of movement per step small enough, the ringing can be completely eliminated at the price of velocity. Results shown here are for a percentage selected such that the total time to move was minimized.

Two-step The motor is moved to the midpoint between the initial and destination positions. After a delay equal to half the period of the natural motor oscillation, the motor is moved to the destination position.

The midpoint and delay used are an approximation to the ideal values. These result in good results with a fixed time to move any distance, and a small amount of ringing at the end of the movement. The ringing at the end of the motion does not depend on the amplitude of the motor movement. Rather, it depends on the inaccuracy of the midpoint and delay calculations. A calibrated look-up-table approach would be necessary to eliminate the ringing completely.

Dual two-step The two-step method is used to move first to the midpoint, and then to the endpoint.

This theoretically takes twice as long as the single two-step method, but actually results in slightly better results because the resultant ringing is somewhat reduced.

Multiple two-step The two-step method is used multiple times, always moving halfway between the current position and the destination position.

This theoretically takes time $O(\log(\Psi))$ where Ψ is the angular distance between the initial and destination positions. In practice, it produced very similar results to the dual two-step method.

Our measuring apparatus was fairly noisy, resulting in the high variance reported. It is clear that all four control strategies provided substantial improvement over the uncontrolled performance. Of these, the two-step methods are a little better than the fixed percentage approach. Because of the noise in our measurements, it is not clear which two-step method is the best.

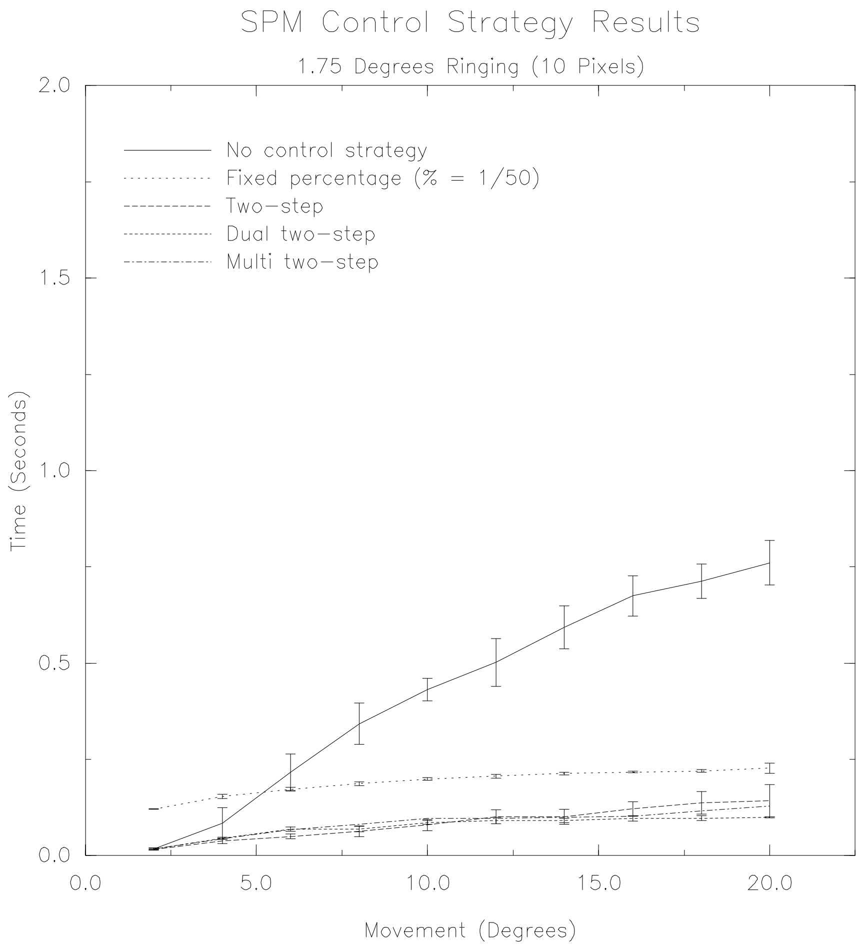


Figure 4.17: Results of different open-loop control strategies to reduce the ringing of the SPM. Times reported are those after which amplitude of the ringing has been reduced to 1.75^o (10 pixels in our camera). Control strategies are discussed in the text.

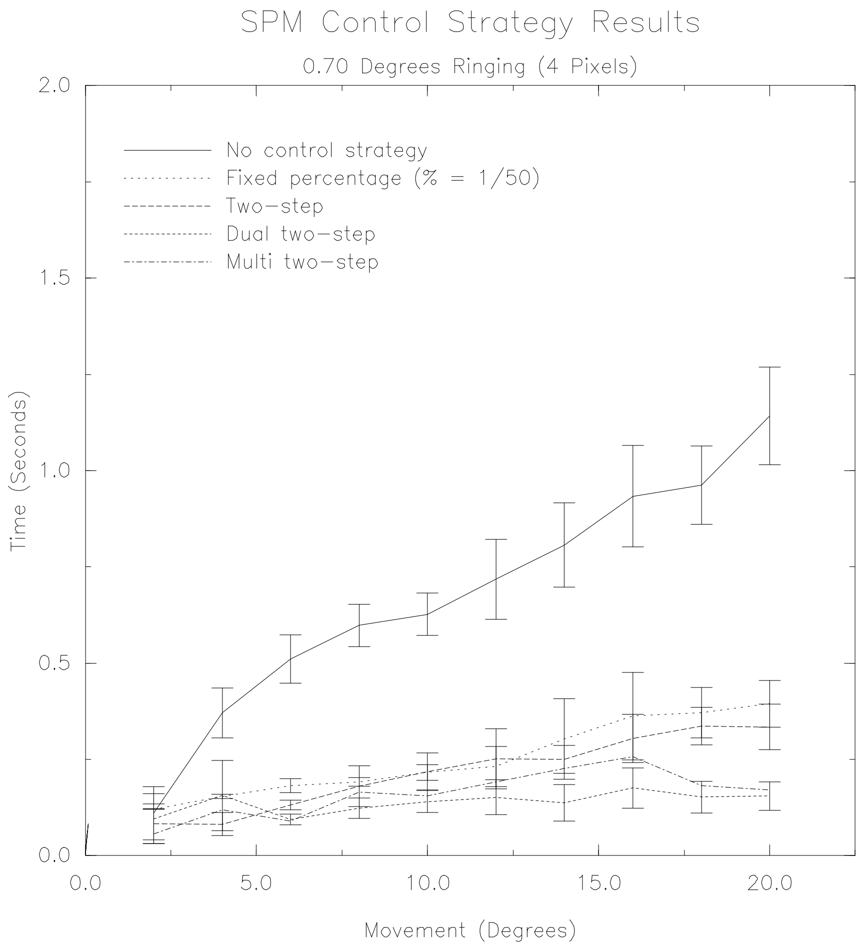


Figure 4.18: Results of different open-loop control strategies to reduce the ringing of the SPM. Times reported are those after which amplitude of the ringing has been reduced to 0.70^o (4 pixels in our camera).

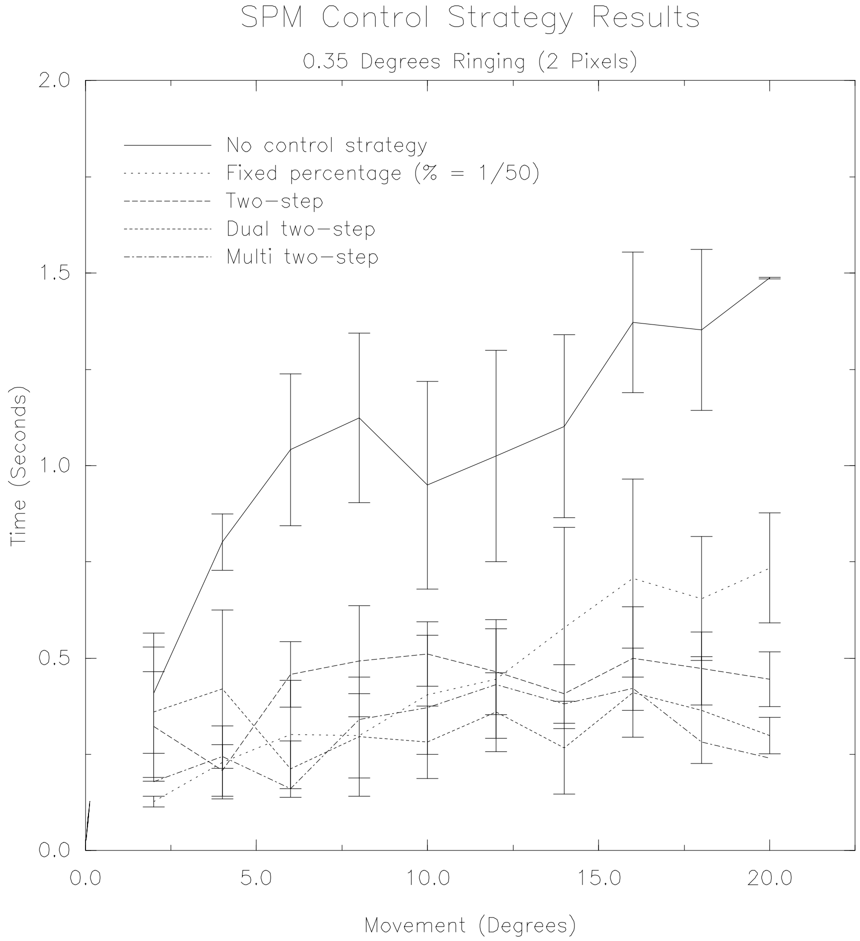


Figure 4.19: Results of different open-loop control strategies to reduce the ringing of the SPM. Times reported are those after which amplitude of the ringing has been reduced to 0.35^{o} (2 pixels in our camera).

The results are summarized in the following table. For each control strategy, the time to move 2 degrees and to move 20 degrees are given for each of three ringing requirements. The times are specified in milliseconds.

Ctrl Method	$.35^o$ ringing	$.70^o$ ringing	1.75^o ringing
No control	408/1480	110/1140	18/760
Fixed percentage	127/734	120/395	120/227
Two-step	332/445	83/334	16/143
Dual two-step	359/298	96/155	18/100
Multi two-step	179/240	56/171	17/128

It should be stressed that the results from these two-step open-loop control strategies were obtained with approximate calculated values for the mid-point and timing values. We fully expect that with a calibrated look-up-table approach, as with the position calibration of Section 4.3, we will obtain substantially better results.