A Real-time 3D Motion Tracking System

Johnny Wai Yee Kam

Technical Report 93-16

April 1993

Laboratory for Computational Intelligence
Department of Computer Science
The University of British Columbia

Vancouver, B.C., V6T 1Z2
Canada

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE

Abstract

Vision allows one to react to rapid changes in the surrounding environment. The ability of animals to control their eye movements and follow a moving target has always been a focus in biological research. The biological control system that governs the eye movements is known as the *oculomotor control* system. Generally, the control of eye movements to follow a moving visual target is known as *gaze control*.

The primary goal of motion tracking is to keep an object of interest, generally known as the visual target, in the view of the observer at all time. Tracking can be driven by changes perceived from the real world. One obvious change introduced by a moving object is the change in its location, which can be described in terms of displacement. In this project, we will show that by using stereo disparity and optical flow, two significant types of displacements, as the major source of directing signals in a robotic gaze control system, we can determine *where* the moving object is located and perform the tracking duty, without recognizing *what* the object is.

The recent advances in computer hardware, exemplified by our Datacube MaxVideo 200 system and a network of Transputers, make it possible to perform image processing operations at video rates, and to implement real-time systems with input images obtained from video cameras. The main purposes of this project are to establish some simple control theories to monitor changes perceived in the real world, and to apply such theories in the implementation of a real-time three-dimensional motion tracking system on a binocular camera head system installed in the Laboratory for Computational Intelligence (LCI) at the Department of Computer Science of the University of British Columbia (UBC).

The control scheme of our motion tracking system is based on the Perception-Reasoning-Action (PRA) regime. We will describe an approach of using an active monitoring process together with a process for accumulating temporal data to allow different hardware components running at different rates to communicate and cooperate in a real-time system working on real world data. We will also describe a cancellation method to reduce the unstable effects of background optical flow generated from ego-motion, and create a "pop-out" effect in the motion field to ease the burden of target selection. The results of various experiments conducted, and the difficulties of tracking without any knowledge of the world and the objects will also be discussed.

Contents

List of Figures

Acknowledgements

Many thanks ...

... to my parents, Mr. Gilbert Sik-Wing Kam and Mrs. Muner Lee Kam, for your love and faith all these years, for providing me an excellent environment to grow up, and for being so understandable and patient during difficult times. This thesis cannot be finished without your support and encouragement.

... to my sister, Elaine, the Pharm.D. to be, my brothers, Timothy, the M.D. to be, and Danny, the software engineer, for everything! Well, what can I say more? I am proud of all of you.

... to Dr. James Little for supervising this thesis, for always coming up with new ideas and problems, and for being such a good friend during my years of studying at UBC. Wishing you and your family all the best, Jim.

... to Mr. Vincent Manis, for your encouragement and effort of making me work so hard for my undergraduate degree.

... to Dr. Alan Mackworth for reading this thesis.

... to Dr. David Poole for teaching me to be *reasonable*.

... to Dr. Robert Woodham and Dr. David Lowe for making Computational Vision interesting and challenging.

... to Mr. Dan Razzell, our LCI manager, for helping me out all these years.

... to Rod Barman and Stewart Kingdon, our LCI technical staff, for setting up the hardware environment, and for providing the *still usable* software.

... to Ms. Valerie McRae, our CIAR secretary, for the caring and support, and for proofreading part of this thesis.

... to my friends, Pierre Poulin, Chris Healey, Carl Alphonce, Mike Sahota, Swamy, Esfandiar Bandari, Stanley Jang, Yggy King, Art Pope, Ying Li, Andrew Csinger, Scott Flinn and Karen Kuder, for valuable ideas, suggestions, and discussion.

... to George Phillips for helping me numerous times with the LaTeXand PostScript problems.

... to all the present and former staff and graduate students at UBC CPSC for making this department a friendly and enjoyable place to work in.

THANK YOU!

Johnny Kam
Vancouver, B.C.
CANADA
April 18, 1993

Chapter 1

Introduction

Vision allows one to react to rapid changes in the surrounding environment. Cues for animals to notice such changes are mostly visual. One obvious change introduced by any moving object is the change in its location with respect to other stationary objects in the environment, where such change can usually be described in terms of displacement, the difference in locations.

It has always been a main focus in research to examine animals' abilities to perceive the changes incurred by moving objects and to react to those changes simultaneously. A particularly interesting area is the ability of animals to control their eye movements and follow a moving target. The biological control system that governs the eye movements is known as the *oculomotor control* system. Within such control system, the two significant types of eye movements are saccade and vergence. The saccadic, or gaze shifting, system enables the observer to transfer fixation rapidly from one visual target to another, while the vergence system allows the observer to adjust the angle between the eyes so that both eyes are directed at the same point. The control of eye movements to follow a moving visual target is generally known as *gaze control*.

Functionally, gaze control allows one to change the direction of gaze from one position to another, and consequently, one can maintain gaze on a chosen target, or in other words, fixate a moving object in the visual system. One can, as a result, gather additional information about such object for further analyses and tasks such as recognition and learning. The cooperation of gaze shifting, gaze holding, and vergence thus allows the task of motion tracking to be performed.

With recent developments in sensors, parallel processors, and special purpose image processing hardware, it is now possible [Little *et al.*, 1991] to attempt to build robotic devices that can simulate an animal's ability to visually track an object moving in three-dimensional space [Ferrier, 1992] [Christensen, 1992] [Jenkin *et al.*, 1992] [Pahlavan and Eklundh, 1992] [Crowley *et al.*, 1992] [Pretlove and Parker, 1992]. The main purposes of this project are to establish some simple control theories to monitor changes perceived in the real world, and to apply such theories in the implementation of a real-time three-dimensional motion tracking system on a binocular camera head system installed in the Laboratory for Computational Intelligence (LCI) at the Department of Computer Science of the University of British Columbia (UBC). The primary goal of this tracking system is to center the image of the object of interest, in this case being an object in motion, as quickly as possible. Such *passive* motion tracking system must be comprised of a module to detect moving objects, a module to select the visual target, and a module to respond in the form of gaze shifting and verging the binocular head. It is also our interest to investigate how different systems, namely, the Datacube MaxVideo system and a network of Transputers, which run at different rates, can communicate and cooperate in real-time.

The detection of moving objects in a scene is difficult when the observer is also in motion, because the dominant motion is usually generated by the moving observer, which leads to a complex pattern of displacements. It is necessary that the system be able to determine objects moving with respect to the stationary environment, rather than with respect to the observer. Some segmentation process must occur to separate the apparent motion, or *ego-motion*, caused by the moving observer, from the motion incurred by the moving objects. The control system is required to stabilize gaze against ego-motion while tracking the moving target.

The whole system, described in this report, will follow the **Perception-Reasoning-Action** (PRA) framework, but with very little representation of the world, or even the observer. Real-time performance is necessary for a real-behaving system [Nelson, 1991]. There is a delay between the time at which some visual observation is made and the time at which the control command based on the observation is computed. To minimize such delays with our limited computational resources, we need to focus on the minimum possible and non-trivial set of visual information necessary to achieve the purpose of motion tracking. The system has to be designed so that responses can be made appropriately and timely in the unpredictable environment, and that it can keep up with the pace of the world.

Researchers have reported that humans have several interacting control systems that stabilize gaze against ego-motion and follow moving targets, but failed to identify the necessary visual cues that should be used in a robotic gaze control system for motion tracking [Ballard and Brown, 1992]. In this project, we show that by using stereo disparity and optical flow, the two significant sources of displacement measures, as primary visual cues in the robotic gaze control system, we can determine *where* the moving object is located and perform the tracking duty, without recognizing *what* the object is. We can easily fixate on a 3D location while ignoring the distracting surrounding motion by integrating stereo with optical flow. In theory, the vergence system can provide the gaze control system with extremely useful input for filtering purposes, and for reducing the volume of space to be considered.

Several researchers have recently been implementing some gaze control systems with encouraging results [Brown, 1989] [Coombs, 1992], and have demonstrated that special purpose hardware are required to perform complicated computation in real-time. As an alternative to using the existing techniques for motion tracking and replicating the work that has already been done, it is our intention to build a system which makes good use of our current specialized hardware, and to carefully allocate resources so that other types of computations can be performed at the same time. It is our belief that if we can *continuously monitor* the changes in the environment accessible to the cameras, we should be able achieve our objectives and compute displacements using a simple correlation matching technique, assuming that such technique can produce reliable and dense data.

Our system uses optical flow and stereo disparity for tracking by panning, tilting, and verging the robot head. An active monitoring process along with the optical flow accumulation processes form our perception system. The reasoning system consists of a cancellation process to eliminate the unstable effects of the background optical flow. A segmentation process is used to partition the flow field into connected components, and allows the visual target to be selected.

Chapter 2 of this thesis describes the research related to the field of motion tracking. The work done on uncovering the mystery behind humans' ability to track moving objects is discussed. Related work on motion detection and integration of stereo and optical flow will be described. Other robotic gaze control systems developed at various research sites, particularly at the University of Rochester, are also discussed.

Chapter 3 describes the PRA model, control theories, and techniques used in this project. The various assumptions made in designing the system will be described.

Chapter 4 contains detailed descriptions about the implementation of our motion tracking system, in terms of both hardware and software.

Chapter 5 presents the evaluation of our motion tracking system. The performance, drawbacks, and various problems will be discussed. We will describe the different experiments that have been carried out, and will also compare our motion tracking system with other types of tracking systems implemented elsewhere.

The conclusions and directions of future work will be followed in Chapter 6.

Chapter 2

Background and Related Work

2.1 The Biological Oculomotor Control System

Researchers have spent considerable amount of effort investigating on the ability of animals, particularly humans, to visually track moving objects. The control system responsible for such trackings directs the eyes to move rapidly to follow a visual target and to stabilize the images of such target on the retina in spite of relative movements between the target and the observer.

The rapid eye movements are known as **saccades**. During these movements, both eyes rotate in same direction. It has been reported that animals do not see well during saccadic movements, and that the oculomotor control system may become unresponsive to stimulus during these movements. Therefore, the oculomotor control system will attempt to make the duration of each movement as small as possible [Robinson, 1968]. A separate system known as the **smooth pursuit** system, which responds to target velocity regardless of target position, operates independently with the saccadic system for stabilizing images on the retina. It is worth noting that the saccadic system is a sampled control system, whereas smooth pursuit is continuous. Researchers have discovered that saccadic movements are made in response to a large burst of tension suddenly applied and suddenly removed, while smooth pursuit movements are created by smaller smoothly applied forces.

Humans possess two eyes mainly to perceive *depth* in the surrounding environment. Stereo disparity serves as a great visual cue to recognizing objects at different depth, and allowing the tracking of objects moving in three-dimensional space. With objects moving near to or far away from the observer, his eyes must make equal movements but in opposite directions, governed by the **vergence** system, in order to keep the target image focused and centered on the retina. When the eyes rotate nasally the movement is called convergence, and when they rotate temporally[1] it is referred to as divergence. It seems clear that a different control system is responsible for vergence movements. Vergence movements allow humans to register an object on the fovea (the central, high-resolution region of the retina) of each eye, so that the greatest possible amount of information about the object can be extracted. Experiments have shown that such system appears to be continuous with a very low gain integrator, which makes it the slowest of all the oculomotor subsystems [Robinson, 1968]. It has also been reported that the disparity vergence system is not only sensitive to the amount of disparity between the left and right retinal images, but also to the rate at which this disparity changes [Krishnan and Stark, 1977].

It is considered that all changes of fixation are made entirely by mixtures of pure saccadic movements and pure vergence movements; that is, the two systems operate independently, and it has been observed that they cooperate in a very complex way.

[1]An anatomical term, meaning towards the temples or the sides of the forehead.

Based on experimental results on stimulations of the brain, voluntary saccades originate in the frontal eye fields, involuntary saccades and smooth pursuit movements in the occipital eye fields, and vergence movements in areas 19 and 22 of the brain. The intersampling interval for saccadic movements is 200 milliseconds, and the time delay is of the order of 150 to 200 milliseconds in the disparity vergence system.

2.2 Detection of Moving Objects

Detection of moving objects from a stationary observer is easy, as the difference of two frames in an image sequence will show the direction and magnitude of the object's motion. A more interesting and difficult problem is to detect moving objects from a moving observer, in most cases being an electronically controlled camera. Being able to find out how much an object has moved with respect to the stationary environment, rather than with respect to the observer, is the key to determining whether or not the object is in motion while the observer is moving.

The dominant motion in the motion field is usually generated by the moving camera, if one assumes that the moving object is much smaller than the background environment in the field of view of the camera. The central idea to identify a moving object involves segmentation of the motion field based on consistency of the pixel values, e.g. the optical flow vectors of a flow image, into separate components. Some robust segmentation algorithms have been presented, such as the one reported in [Adiv, 1985], where the flow field is partitioned into consistent segments, and independently moving rigid objects can be analysed. A main interest in recent research is to identify which components correspond to moving objects, and which portions of the motion field are caused by the moving observer, with or without knowing the ego-motion parameters.

Gibson makes an interesting observation that the flow vectors due to all stationary components intersect at the *focus of expansion* (FOE) [Gibson, 1979]. Moving objects have their individual FOE, usually different from the FOE due to the stationary component. Jain describes how an ego-motion polar transform of the dynamic scenes acquired by a translating observer can make moving objects easily distinguishable from stationary ones [Jain, 1984]. However, such technique requires intensive computation for the transformation and the recovery of the FOEs, and is rather inefficient to use in any real-time system.

Thompson and Pong describe the principles of detecting moving objects under various circumstances [Thompson and Pong, 1987], but leave an open question on how to apply such theories in practice. Different assumptions are made under different situations, and various detection algorithms are developed. In particular, situations where the camera motion, either a translation or a rotation, is known or unknown are examined. Point-based, edge-based, and region-based techniques which relied on knowing the FOE are applied, and the techniques have been shown to be quite robust against noise. They further state that no method for detecting moving objects will be effective if it depends on knowing precise values of optical flow. However, it is clear that the effectiveness of any reliable technique is directly proportional to the preciseness of the input data. Moving objects can be incorrectly identified if the input data is not reliable, and thus the motion detection process becomes impractical.

Another observation made during these studies is that *if an object is being tracked, its optical flow is zero* [Thompson and Pong, 1987]. This phenomenon can be seen in the ideal situation where an observer is actively tracking a moving object, and knows ahead how far the object will move courtesy of a prediction system, so that the optical flow of such moving target being followed is effectively zero. But this observation cannot be made easily in a motion tracking system when the motion parameters of the camera is a function of how far the object has moved in a certain time interval. In other words, if we have a *passive* system whose actions are triggered by changes that are occuring in the real world, then it is unlikely that the passive system can maintain the tracking operation on a moving object without non-zero optical flow input. However, the observation still holds if one is to consider that the optical flow of the moving object produced is the displacement of such object with respect to the image plane before the motion has started and after the motion is completed.

Some subspace methods and center-surround motion operators for recovering observer translation [Heeger *et al.*, 1991] and a least squares method to solve the *brightness change constraint equation* for the motion parameters in the case of known depth [Peleg and Rom, 1990] have been introduced. These methods, however, may not be suitable to apply in a real-time system due to the fact that the motion parameters must be recovered before different components can be identified and that extensive amount of computation are usually required.

The work by Nelson addresses the problem of identifying independently moving objects from a moving sensor [Nelson, 1990]. Two methods are discussed, one making use of the information about the motion of the observer, and the other using knowledge about how certain types of independently moving objects move. The implementations that run in real-time on a parallel pipelined image processing system are described. The first method, called *constraint ray filtering*, is based on the idea that in any rigid environment, the projected motion of any point is constrained to lie on a one dimensional locus in the velocity space whose parameters depend only on the observer motion and the location of the image point. An independently moving object can be detected because its projected velocity is unlikely to fall on this locus. The restriction is that observer's motion has to be known prior to the computation. The second method, known as the *animate motion method*, uses the idea that the observer's motion is generally slow and smooth, whereas the apparent motions of independently moving objects are comparatively changing more rapidly. The limitation of this method is that it is insensitive to smoothly moving objects. These methods are in general quite resistant to noise, and can make use of motion information of low accuracy. Both methods have been successfully implemented on the Datacube MaxVideo system, and an update rate of 10 Hertz is achieved. Our system is like Nelson's since it combines motion, real-time processing, and known parameters of the observer's motion.

Some researchers have recently said that it is easier to detect tracking signals in active visual following than in passive tracking, as motion blur emphasizes the signal of target over the background [Coombs and Brown, 1992].

2.3 Integrating Stereo and Optical Flow

Stereoscopic motion analysis combines stereo data and motion data computed from the binocular images sequence. It relies on the *dynamics* of the scene. Stereoscopic image analysis usually requires images to be taken from the same scene at the same time from two parallel viewing points lying on a horizontal line. Matching techniques have been developed to compute stereo disparities from the left and right images pair, as in [Marr and Poggio, 1976] [Drumheller and Poggio, 1986] [Fua, 1991], so that the 3D locations of the image points can be recovered providing that the geometry of the stereo cameras configuration is given. Real motion in the 3D world projects to 2D images to produce apparent motion in the image plane known as optical flow. A simple correlation method can be used to compute optical flow, the 2D motion vector. A reliable coarse-to-fine matching technique has been developed in [Anandan, 1989]. Some simple, efficient, and robust parallel motion and stereo algorithms, as presented in [Bulthoff *et al.*, 1989]

[Little and Gillett, 1990], can also be used.

The 3D motion vector for each image point can therefore be computed using both stereo and optical flow information. The 3D motion parameters, with respect to a world coordinate system, can then be recovered for a corresponding group with consistent motion vectors.

Previous work in combining stereo and optical flow was mostly aimed to either solve the correspondence problem [Waxman and Duncan, 1986], or to recover the motion in depth parameters [Balasubramanyam and Snyder, 1988]. Stereo data provides *absolute* depth information as additional constraints. These constraints will likely reduce the computation needed as compared to the situation where the motion parameters are to be determined without the depth, even though it has been shown that information about depth, structure, and motion of objects relative to the observer can be determined from optical flow alone. Numerous experiments have also confirmed that a binocular camera system provides a more robust sensing mechanism while operating under realistic conditions. Some work on extracting 3D motion and structural data illustrates that it makes more sense to work in 3D but at the cost of being very sensitive to noise [Zhang and Faugeras, 1992].

2.4 Active Vision, Behavioral Vision, Animate Vision, and Passive Vision

An important idea in current computer vision research is that the vision process is *dynamic* rather than static [Clark and Ferrier, 1988] [Ballard and Brown, 1992]. As a result of various work done by different research groups, several coherent themes have emerged. The use of active sensing to continuously gather information has received considerable attention. Issues on how to react to the rapid changes perceived and how to control the visual sensors have been the main foci in most studies.

Active vision, as considered by most researchers, refers to the process of making a vision problem that is ill-posed into a well-posed one, or in other words, converting an underdetermined problem into an overdetermined one by changing the parameters of the visual observer. The two common tasks of any active vision system are to figure out where to look next, and to carry out the motion that will let one look there. The class of active vision algorithms shares the idea of using an observer actively providing constraints that can simplify computation of image features, and help eliminating ambiguities. Active vision algorithms can be more robust than static algorithms, and are often computationally efficient since irrelevant information is ignored, so that the process for finding a solution is easier and the results are more reliable.

In recent years, research at the University of Rochester has reflected the theme that understanding the phenomenon of intelligence and discovering how to produce an artificial one must proceed in the context of *behaviour* [Nelson, 1991]. Behavioral approach to AI, vision in particular, has received considerable attention as it is observed that most ideas for machine intelligence are inspired by the abilities of animals, particularly humans. One important strategy that has been used is to throw out as much information as quickly as possible, since the total quantity of information contained in a visual signal often exceeds the capacity that any system can handle. The vision system has to keep up with the pace of the world, and it does not compute all things at all time, but only what it needs at a certain time. As a result, the amount of representation that is required may be drastically reduced, thus freeing up the valuable limited computational resources.

To emphasize the focus on the human-like aspects of vision and control schemes, the term *animate vision* was introduced. Animate vision is a framework for sequential decision making, gaze control, and visual learning [Ballard and Brown, 1992]. As stated in the report, the interactionist approach is based on the idea that the world and the perceiver should participate jointly in computation, and that neither is complete without the other. For instance, the world can be viewed as an external memory, where gaze control positions the eyes appropriately at the point of application to allow decisions to be made.

Prediction and perhaps learning are the most important ingredients that distinguish active systems from passive ones. If the goal of an active vision system is to track a moving object, it may have to first actively explore the environment to *look* for a specific target to follow, with or without recognition. A prediction engine will cooperatively command the observer to track the target, in addition to the signals returned by the motion detector. The behaviour of the visual target learned through such active sensing helps to strengthen the reliability of the responses made in the unpredictable environment. In passive systems, however, the observer's motion is solely the reaction to the changes perceived. The importance of this type of reactive responses is captured by Brooks' subsumption architecture, and his idea of using very little representation when behaviours are taken to be the fundamental primitives [Brooks, 1987].

2.5 Gaze Control

Humans have several interacting control systems that stabilize gaze against ego-motion and follow moving targets. Recent research in gaze control mechanisms has been mostly to replicate this important behaviour in a real-time computer controlled environment. Due to the fact that the processing is inevitably computationally intensive, parallel computer systems and specialized image processing hardware are needed. However, the strategies used for gaze control must be cooperative with the hardware and efficient enough so that the system can interact with the world and has active control over its own state in a timely and consistent manner. Gaze control is made up of two subproblems: gaze holding and gaze shifting. The main concern is with *gaze holding*, the operation of maintaining fixation on a moving object with the cameras on a moving platform.

There are a few reasons why we need gaze control. Functionally, we want to change the direction of gaze from one position to another, and to fixate an object to minimize motion blur. Being able to fixate an object allows the observer to gather more information for further analyses. In theory, it has been pointed out that the object of interest should be kept at the center of the images to obtain higher precision, and that the greatest amount of information can be extracted. In addition, segmentation is known to be a problem as it is difficult to separate an object from the background in a scene without recognition, but it is hard to recognize the object without separating it from the background. Gaze control can help to solve this "chicken-and-egg" problem by separating a moving object from background without recognition, based on the idea that stabilizing one point in the scene that is moving relative to the observer induces target "pop-out" due to motion blur induced in the non-stabilized parts of the scene [Ballard and Brown, 1992].

The three types of controls which are common and necessary in a robotic camera head, or Eye-Head, system are panning, tilting, and verging.

Panning refers to the process of rotating the inter-camera baseline about a vertical axis. The pan motor thus will move the head in the horizontal direction.

Tilting refers to the process of rotating the inter-camera baseline about a horizontal axis. The tilt motor will be responsible for vertical motions.

Vergence is the process of adjusting the angle between the eyes, or cameras, so that both eyes are directed at the same world point. It is an antisymmetric rotation of each camera about a vertical axis.

With these three degrees of freedom, one can theoretically place the intersection of the optical axes of the two cameras anywhere in the three dimensional volume about the head.

2.5.1 Real-time Binocular Gaze Holding System at Rochester

David Coombs and his colleagues have successfully designed and implemented a gaze holding system on a binocular camera head system at the University of Rochester [Coombs, 1992]. Their work has focussed on the problem of using visual cues alone to hold gaze from a moving platform on an object moving in three dimensions. The system implemented on the Rochester head demonstrates that gaze holding can be achieved prior to object recognition, assuming smooth object motion. The vergence and pursuit systems perform complementary functions in the sense that the pursuit system centers the target, and the vergence system converges on it. Interestingly, the vergence system minimizes the disparity on the target being foveated, and the pursuit system requires that the target be properly verged before it can locate it and center it on the coordinate system.

The vergence system estimates vergence error based on stereo disparity. Disparity is measured with a *cepstral filter*. The cepstrum of a signal is the Fourier transform of the log of its power spectrum, and the power spectrum is just the Fourier transform of the autocorrelation function of the signal [Olson and Coombs, 1991]. The disparity estimator will report the disparity that bests accounts for the shift between the images, in this case being the tallest peak in the spectrum. The control system will then generate smooth eye movements to correct the vergence error.

The pursuit system attempts to keep the visual target centered in the cameras' images, assuming that the vergence system can keep the cameras verged on such target. The visual target will be the object that is both near the center of the current image and in the region called *horopter*, which is the 3D locus of points with zero disparity at the current vergence angle. The core of this subsystem is the Zero-Disparity Filter (ZDF) [Coombs and Brown, 1992]. This filter does not measure disparity, but it locates the portions of the images that have zero stereo disparity. It is a non-linear image filter which suppresses features with non-zero disparity, and can be implemented in the real-time system using the *total mask correlation* method [Coombs, 1992]. The windowed output of this filter will be the input to the pursuit system, which guides the cameras to pan and tilt so that the object, with zero disparity, will be centered on the image.

Various control techniques have also been implemented to generate smooth camera movements. In particular, the α–β–γ predictor, a linear Kalman filter, is used to smooth the verging angles, and to smooth and interpolate the target positional signal, assuming that the target signal has uniform acceleration [Coombs, 1992]. The $\alpha-\beta-\gamma$ predictor is also used to predict delayed signals, which can lead to more accurate tracking.

Delay can cause a system to be unstable. The two major factors causing delays are computation and transmission. Researchers at the University of Rochester have used Smith prediction and multiple Kalman filters to cope with delays, and the results have been satisfactory.

Chapter 3

Proposed Techniques and Control Theories

3.1 Objectives, Purposes, and Assumptions

The main objectives of this project are to establish some simple control theories to monitor and to react to changes perceived in the real world, and to apply such theories in the implementation of a three-dimensional motion tracking system on the robot head installed in LCI. The configuration of this LCI binocular camera head system will be described in the next chapter. The primary goal of our tracking, or gaze control, system is to center the image of the object of interest as quickly as possible. Such a passive motion tracking system must be comprised of a module to detect moving objects, a module to select the visual target, and a module to respond in the form of gaze shifting and verging the binocular head.

Displacement is perhaps the most explicit and direct form of measurement to represent the change in locations of an object that appears in the images input through a camera. **Optical Flow** is the motion displacement which allows the determination of how far the object has moved, usually during a short time interval. **Stereo Disparity** is the displacement measure which shows the difference in the relative locations of a point registered in the left and the right stereo images grabbed at the same time. It is worth noting that stereo disparity is inversely proportional to the *depth* measured with respect to the cameras. Our goal is to find out how to make appropriate use of optical flow and stereo disparity to perform tracking in a real world situation, and to stabilize the cameras' movement.

It is also our interest to investigate how different computer systems, namely, the Datacube MaxVideo system and a Transputer network, which run at different rates, can communicate and cooperate in a real-time environment. The motion tracking system has to be designed so that responses can be made correctly and timely in the unpredictable environment, and that it can keep up with the pace of the world. The demonstration systems implemented in this project to illustrate our theories are quite simple, but they do form, in our opinion, the basis of more sophisticated motion tracking systems.

Our control theories are designed to be as general as possible, but it is inevitable that *assumptions* have to be made as there are many uncertainties and exceptions which are very difficult to handle without using special or complicated procedures. The following is the list of assumptions that we will undertake in our theories:

1. Much of the vision research to date is concerned with rigid objects, since non-rigid objects are very difficult to handle without prior knowledge about their behaviour. Therefore, to keep this project tractable, we will assume that we are only dealing with the motion of rigid objects.

2. We will assume that displacement measures are being returned continuously in

a frame-by-frame manner. This mechanism is analogus to the natural ability of humans to detect changes caused by a moving object in the way that the changes can be *seen* in a continuous fashion.

3. The technique used in measuring displacements is fast and accurate, and that *dense* and *reliable* integer valued stereo disparity and optical flow values are always produced.

4. The camera geometry is known. In particular, the *focal length* and the *baseline separation* of the stereo cameras used should be provided.

5. We will assume that the motion of any object is *slow*, and *smooth* enough to work with. This assumption reveals the fact that every system has its limits, and it is unreasonable to expect our motion tracking system to work in all scenarios, such as following a high-speed bullet.

6. The motion of the observer will also assume to be slow, and smooth. This allows the measuring technique to work on a smaller range of possible matches.

7. The motion tracking system is expected to be run on some special purpose image processing hardware and a multi-computer system, so that the control procedures can be designed to take advantage of the powerful computational resources.

8. We assume that the motion parameters of the observer are known in the entire system, or available upon request.

9. For the sake of simplicity and ease of illustration of our ideas, we assume that optical flow is computed from the viewpoint of only one camera, presumably the left camera.

10. We will not have any other specific knowledge of the objects we are dealing with.

3.2 Perception-Reasoning-Action Control Scheme

The perception-reasoning-action (PRA) control loop is commonly used in traditional AI problem solving techniques. In this control strategy, three subsystems are generally used: the **perception system** provides input to the **reasoning system**, and the task to be performed by the **action system** depends on the output of the reasoning system (see figure 3.1). PRA loop is interesting and popular due to the fact that it is a fundamental concept which is easy to understand. It even resembles one of the ways humans solve problems. In the human visual system, the visual input is often analysed without conscious attention before we proceed with our actions. This phenomenon can easily be found in many of our activities such as driving a car, playing video games, or visually following a flying plane.

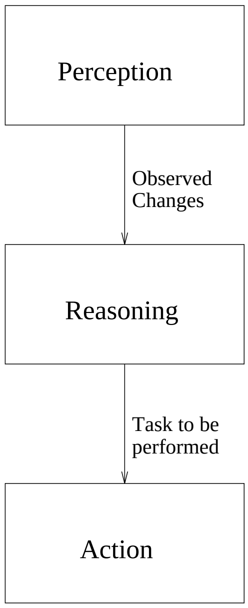

Figure 3.1: PRA Communication

Our control theories will follow the PRA framework. Changes in the environment will be observed from a pair of stereo cameras by the perception system. The reasoning system will process the changes represented by optical flow vectors and stereo disparities, find out the new location of the moving target, and provide information for the action system to track the target by physically moving the robot head. Although the data flow appears to be sequential within the loop, the three subsystems are actually *parallel processes* running concurrently. For example, the perception process will continue to monitor the changes while the reasoning processes are busy analysing the data. Other processes can be actively communicating with one another at the same time.

3.2.1 Perception

The role of the perception system is to pay attention to the changes in the environment. Changes can be computed and returned as an image in minimal time by powerful image processing hardware. As a result, a continuous flow of these images will be produced for further processing. In consideration of the delays that are coupled with the computation in the reasoning process, special control procedures are needed to ensure that these images would not be lost and that all data would be available when the reasoning process needs the data. We introduce the idea of using an *active monitor* and an *accumulation process* to keep track of those data provided by the optical flow images. As a consequence, we are observing the world and monitoring the changes in a continuous fashion, without any representation of the world, the objects[1], and even the observer. Besides, the perception system does not play an active role in passing data to the reasoning system once they are available. In fact, the transfer of data is demand driven,

[1] We are referring to special structural representation for recognizing objects. It should be obvious that optical flow vectors represent the locations and sizes of the moving objects, but not the shape, color, or other recognizable features.

but the perception of changes is always active.

The perception system is composed of three subsystems, each being a parallel process communicating with the other process through message passing (see figure 3.2).

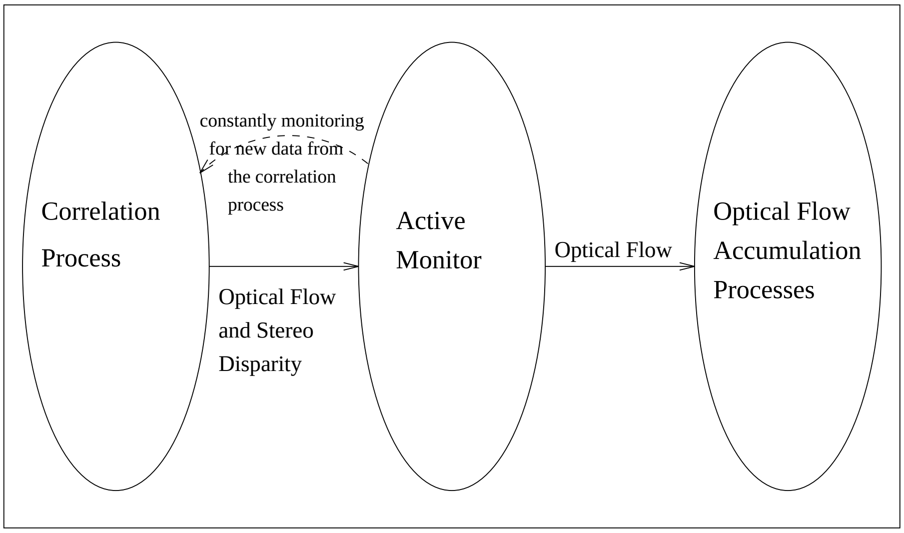

Figure 3.2: The Perception System

3.2.1.1 Computing Optical Flow and Stereo Disparity

Our control strategies rely on the assumption that reliable displacement measures can be produced rapidly. It is anticipated that the displacements computation would be performed on high-speed computers, so that results are available at a reasonable rate. A simple correlation technique, such as the *sum of squared differences* (SSD) or the *sum of absolute values of differences* (SAD) algorithm, can be used effectively because the range of motion is expected to be small with the slow moving objects, and the range of disparity is not expected to be large in one dimensional space. Some correlation matching techniques have been shown to be capable of producing dense flow maps at an acceptable level of accuracy [Fua, 1991] [Anandan, 1989] [Barron *et al.*, 1992].

Correlation should be performed on the whole image. For every point of the image, an array of correlation scores can be computed by taking a fixed window in the first image, and a shifting window in the second. The pixel that has the best match is the one with the optimal correlation score, most likely the minimum number if SSD is used. A verification process can be used to ensure that data returned is reliable, providing that any extra computation would not seriously degrade the overall performance and response time.

It should be pointed out that the more frequently the optical flow image is returned, the more changes the perception system should be able to capture from the real world. This observation can be made when trying to compare the absolute changes captured by x frames versus $2x$ frames returned by the matching operations at a fixed time interval. Given the fact that the size of the correlation window will not change during such operations, the $2x$ frames sequence should contain more accurate information about the motion of the moving objects.

3.2.1.2 An Active Monitor

Our design of the motion tracking system has taken into consideration that different programs running on different architectures have to communicate and cooperate in real-time. In the likelihood that the processes for computing and returning optical flow and stereo disparity will be running on some special purpose image processing hardware, other programs, possibly running on different platforms or configurations, attempting to work on such returned data should employ a process responsible solely for receiving the data. This frame grabbing process, which we denote as the *active monitor*, waits for data to come in, and keeps track of the data until it has been properly stored for later retrieval by the reasoning system. This observing or monitoring process is termed active since it is an independent process which has the initiative to grab a frame whenever it is available without having to wait for instructions to do so.

This particular observing process is simple but plays an extremely important role in synchronizing the communication and uniting the different modules. Optical flow and stereo disparity images will be pumped out continuously regardless of whether or not the reasoning system is prepared to process the data. It is extremely important that this observing process be executed at a rate *no slower* than the rate the displacement measures are being pumped out, or otherwise, the loss of any data frame might greatly contribute to the inaccuracies of the overall system. This is especially critical since we accumulate displacements over several frames.

3.2.1.3 Optical Flow Accumulation

When using parallel processes running on some multi-rate systems to perform motion tracking, there is usually no guarantee that a recipient process, in most cases being a process in the reasoning system, can be freed to receive and work on the displacement data once they become available through the production of the correlation system. However, it is unacceptable, in a real-time system, for any process to block itself without carrying on with its own duties simply because some recipient processes are in busy states and are not ready for the data. The active monitor provides the perception system an opportunity of paying attention to all displacement data produced by the correlation system during one cycle of the PRA loop. Such data must then be properly stored and managed so that it can still be accessible at the appropriate time.

The two different types of displacement measures used in this motion tracking system have different characteristics. Optical flow can be considered as temporal data as it represents how and how much an object has moved within a certain time period. A sequence of optical flow images can thus provide the motion path over a lengthy period of time. Stereo disparity gives the observer an idea of how close an object is at a particular time instant. Although one can in general detect moving things from a sequence of stereo disparity images, extra effort for matching is required, and is usually not desirable if optical flow is already present.

Knowing the fact that optical flow vectors represent changes in locations over time, we suggest adopting the idea of *accumulating* optical flow vectors for storing such changes in order to make them accessible for later uses. The accumulation is basically simple vector addition, and it works as follows:

Let $OF^{P_x}_{t_0,t_1}$ be the optical flow vector representing the change of location of a point P_x from time t_0 to time t_1.

Similarly, $OF^{P_x}_{t_1,t_2}$, $OF^{P_x}_{t_2,t_3}$, ..., etc, will be available as time goes on.

We can compute the absolute change of location of such point P_x from time t_0 to time t_3 by adding the three vectors $OF^{P_x}_{t_0,t_1}$, $OF^{P_x}_{t_1,t_2}$, and $OF^{P_x}_{t_2,t_3}$, i.e.,

$$OF^{P_x}_{t_0,t_3} = OF^{P_x}_{t_0,t_1} + OF^{P_x}_{t_1,t_2} + OF^{P_x}_{t_2,t_3}$$

In general, (referring to figure 3.3)

$$OF^{P}_{t_0,t_n} = OF^{P}_{t_0,t_1} + OF^{P}_{t_1,t_2} + ... + OF^{P}_{t_{n-1},t_n}$$

It should be pointed out that several of these accumulation processes can operate concurrently on different parts of the flow field, so that time delays due to accumulation can be minimized.

As a result of accumulation of optical flow vectors, the *absolute changes* in 2D locations of the moving objects during one cycle of the PRA loop can be safely stored until

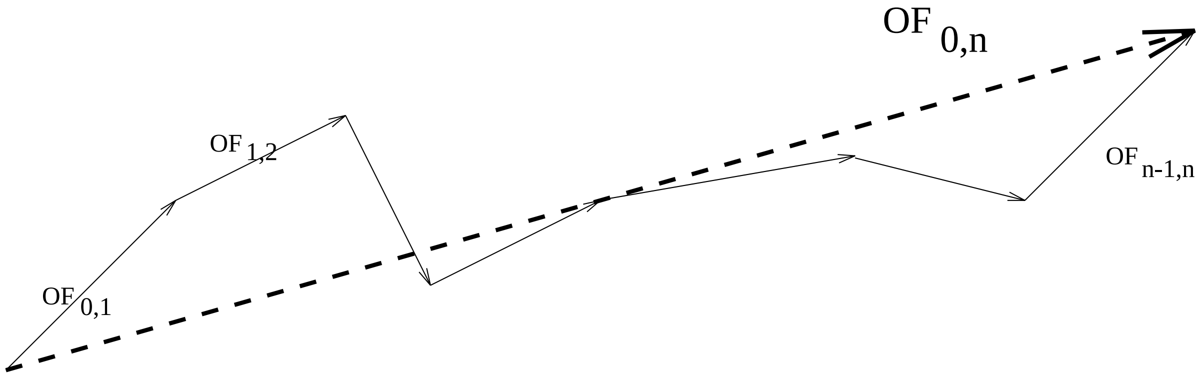

Figure 3.3: Optical Flow Accumulation

such data is demanded by the reasoning system. Stereo disparity data does not require any special procedure to manage, as we are only interested in knowing the depth of an object at a particular time instant. The 3D motion path of a moving object can easily be constructed even if we pay attention to the depth only at the beginning and ending of one PRA cycle.

By adapting these simple ideas of using an active monitor and the accumulation of data, our motion tracking system with different modules running on various computer systems can communicate without having to worry about the timing and synchronization problem. In addition, simple but reliable correlation matching techniques can be used instead of complicated matching procedures. The determination of optical flow and stereo disparity will be a continuous process, instead of being demand driven, i.e., matching is performed automatically at all time, as opposed to having the matching process activated by the reasoning system for request of data.

3.2.2 Reasoning

The main objectives of the reasoning[2] system are to analyse the accumulated optical flow data along with stereo disparities, and to pick out the object to be tracked. The input to this reasoning system is obviously the displacement measures provided by the perception system. Data will be available on request in this interactive parallel environment (see figure 3.4).

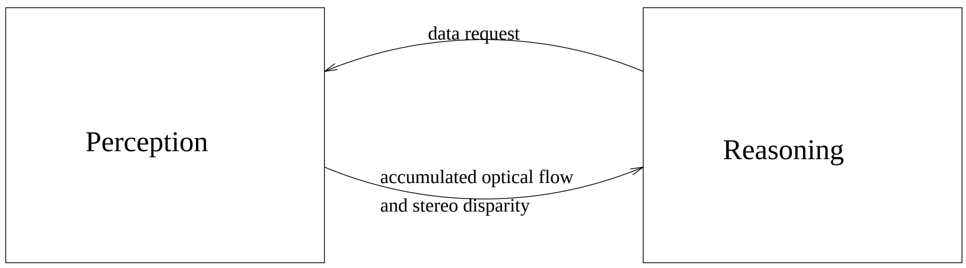

Figure 3.4: Data Transmission between the Perception and Reasoning Systems

Three procedures are used in our design to select the visual target to follow. The background optical flow caused by the motion of the robot head will be taken care of by a cancellation process. The revised accumulated optical flow will be used by a segmentation process to find the different connected components. The object of interest can be chosen among these components as either the region with the largest area, the region with the highest velocity, the region which is closest to the camera with a reasonable size, or various combinations of the above.

[2]One might argue that there isn't much reasoning, as in logical reasoning, involved in this motion tracking system. The term "reason" used here is in the context of attempting to generate some proofs to justify the actions to be performed, and hence it is more than just simple perception.

3.2.2.1 Cancelling Background Optical Flow Caused by Ego-Motion

During each cycle of the PRA loop, the reasoning system is responsible for analysing the accumulated optical flow produced by the perception system before instructing the action system to move the robot head. During such time delays of processing the displacement data and moving the head, objects will continue to move in the real world and the accumulation of optical flow will continue to operate as a separate parallel process. Not only does such accumulation record the changes caused by moving objects, it also records the background optical flow caused by the moving robot head.

The dominant motion in the motion field is usually generated by the moving cameras, if one assumes that the moving object is much smaller than the background environment in the field of view of the cameras. The background optical flow caused by ego-motion makes the job of detecting moving objects much tougher than with stationary cameras. The background flow perceived obviously travels in the opposite direction of the moving cameras. In theory, the whole scene with respect to the cameras will shift during the cameras movement, regardless of whether or not there is any moving object in view, but with the expection that objects moving in the exact same speed and direction with the cameras will appear stationary in the image sequence.

The magnitude of the background optical flow is a function of depth with respect to the cameras and the ego-motion parameters. Closer objects appear to have a larger background flow than farther objects. This phenomenon certainly leads to an extremely complex pattern of flow field for analysis. To compensate for such apparent motion of the background caused by camera motion, we suggest using a method of *cancellation* to reduce the effect of the background flow with the aid of stereo disparity, and the known motion parameters of the cameras.

As part of the initialization process of our motion tracking system, a table of depth to flow value mapping figures will be established. This *special mapping table* can be indexed by stereo disparity, and it contains how much background optical flow is expected to be perceived if the cameras move in one degree in certain direction at a particular depth value, for instance, one degree of panning to the right with stereo disparity of +2 may generate a -3 flow value. Separate entries for panning and tilting motions are required.

This table can be constructed using the following simple procedure. Assuming that all the depth values the correlation system can handle are represented by objects that are within the reach of the cameras. We also assume that all objects will be stationary during this process of initialization. For each disparity value within the correlation range, we first find out a representing point in the image that has the particular disparity value, or depth, as the reference point for determining the motion displacement. The next step is to move the cameras at a known but small angle, and then perform correlation on the images taken before and after such movement. Several repeating moves with different move angles may be required in order to get better approximations of the mapping values.

Given the facts that we know how much the cameras have moved during one cycle, and that we have access to the stereo disparity data before and after such cameras' movement, we are able to first determine the depth of any point in the motion field, and then compute the **expected background flow** for that point using the special mapping table.

Cancellation is therefore a simple procedure of *subtracting* the expected background flow from the motion field. In the ideal situation, expected background flow should be equal to the apparent motion perceived during the accumulation process. However, this is not always the case as time delays for both accumulation and correlation fail to ensure that the accumulated optical flow would contain the exact changes that have occurred in the real world. Also, any round-off error in computation is always a factor causing such inequality. Nevertheless, cancellation creates a "pop-out" effect in the sense that although the background optical flow cannot be eliminated completely, its unstablized effect should be drastically reduced. In other words, the optical flow incurred by any moving object should be significantly larger than the background flow after cancellation. As a result, the revised accumulated optical flow field should reflect the *correct*[3] displacements that the moving objects really caused.

3.2.2.2 Segmenting the Optical Flow Field into Connected Components

The next step of the analyses is to segment the revised accumulated optical flow field into various connected components. A common definition of an *object* encountered in a segmentation algorithm is that it is a set of tokens with the same kinematic parameters [Zhang and Faugeras, 1992]. By applying such concept to our system, an object can be a set of connected points with the same or extremely close optical flow features. A connected component can therefore be interpreted as the group of motion vectors that corresponds to an independently moving object in the scene.

Optical flow based segmentation methods have all the drawbacks associated with the computation of optical flow. It is our hope that the correlation errors can be minimized by the smooth motion constraint. Any segmentation algorithm which is fast and reliable can be employed in our motion tracking system.

Adiv's approach for segmentation first partitions the flow field into connected segments, and then groups the segments under the hypothesis that they are induced by a single, rigidly moving object [Adiv, 1985]. Such technique has been shown to be relatively reliable, but the extensive computation involved makes it an undesirable candidate for use in a real-time system. Similarly, Jain's technique for segmentation [Jain, 1984],

[3] Or an extremely close approximation.

which is based on consistency of the focus of expansion, also requires large amount of computation.

A simple *labelling* or tagging algorithm based on 4-connectedness on a 2D plane has been used in our implementation to partition the flow field using optical flow features. Such algorithm will be described in the next chapter. A region is *4-connected* if every 2 pixels can be joined by a sequence of pixels using only up, down, left, or right moves.

Optical flow vectors corresponding to a moving object may not be equivalent due to the reason[4] that structural differences of the surfaces of an object cause the vectors to vary slighty, or that noise in the images or errors in correlation matching contribute to the inaccuracies of the vectors. In either case, *thresholds* ought to be used in the segmentation process to ensure that vectors should be grouped together if their features match to a certain degree. Such features, for determining whether or not a vector should belong to a region, must be unique or have strong potentials in classifying the flow vectors. The *magnitude* and *direction* of a vector are good candidates for defining connectedness in any vector segmentation algorithm. Two vectors can be defined to belong in the same region if the differences of their magnitudes and directions are less than some pre-defined thresholds.

The output of this segmentation process is a list of connected components, each being described by its location, size, magnitude of the average flow, average disparity, and possibly other symbolic descriptions. A selection procedure is then required to pick out one of these components as the visual target.

[4]Another well-known cause is the *aperture problem*, that is, the image data may not be sufficient to determine the optical flow to more than just a linear collection of velocities.

3.2.2.3 Picking the Visual Target

The input to this procedure are the connected components computed by the segmentation process, and the stereo disparities produced by the perception system. No object recognition or any other features extraction process has been or will be used. This selection procedure must work on what is given as quickly as possible, so that responses can be made in real-time. Such procedure must also take into consideration that multiple moving objects can present at the same time, and there have to be some criteria for picking a specific one out.

There are several options as to which connected component should be picked as the visual target. Some of these options are:

- picking the region with the largest area
- picking the region with the highest average velocity
- picking the region which is closest to the camera
- using a combination of the above suggestions

Picking the region with the largest area without paying any attention to the velocities is perhaps the most unstable method. It has been pointed out that the cancellation of background optical flow does not guarantee to zero out all background flow because of round-off errors and errors in correlation matching. The largest connected component can be the region containing the remains of all the background optical flow vectors, and this is definitely not a target we are looking for.

One obvious choice for eliminating the picking of such background region is to pick the region with the highest average velocity. This idea is inspired by the "pop-out" effect caused by cancellation. Although it is likely that the background region would never get picked, noise is the major factor why this method is not robust, as the region with highest average velocity can be one that is created by noise or errors.

More constraints will undoubtedly make the selection task much easier. Using stereo disparities will give the connected components a three-dimensional look. In other words, these components will be further segmented or layered in the new dimension (see figure 3.5).

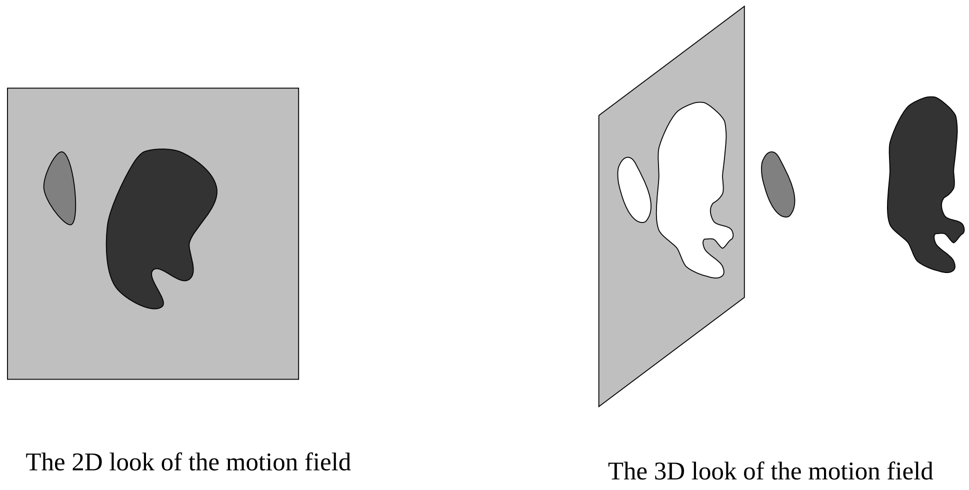

Figure 3.5: The Two-Dimensional and Three-Dimensional Looks of the Optical Flow Field

Picking the region which is closest to the cameras is not a bad idea, if one considers that the motion tracking system is one that should always be alert to attackers. However, noise is again a big factor in disabling this method to work to its full potential.

A slightly different idea is to pick the region which is closest to the zero-disparity surface imaginatively created by the two cameras [Olson and Coombs, 1991]. If our goal of motion tracking is to simply follow the moving object by panning, tilting, and also verging the robot head, then this is probably one of the better selection methods. However, only regions with a reasonable size should be picked, if we want to eliminate the noise problem, but it is rather difficult to define such condition based on what is given. It is also an open question as to whether or not velocity should be a factor in this selection method.

Since we are assuming that the system has no knowledge about any moving object, *perceptual grouping*, the property of human visual perception to perceive discrete blobs moving together as a single object, cannot be modelled in our motion tracking system. Such deficiency is certainly a big factor causing errors in the location of an object during tracking, in the case where the motion field corresponding to the rigid and contiguous visual target is split into several pieces because part of such object is being occluded. Some model-based region growing techniques, e.g., [Shio and Sklansky, 1991], have been presented to solve such problem, but it is beyond the capacity of our system for the reason that object models must be used.

Picking the right region to track is perhaps a more difficult and ambiguous task than initially imagined. It certainly relies heavily on how the motion tracking system is defined to behave, and since there are many different ideas that can be explored, there is definitely no single solution which would work in all scenarios. Perhaps experimentation can provide some guidelines as to how a *confidence measure* can be computed to decide which method is best and more robust to use.

3.2.3 Action

The primary goal of the action system is to respond to the locations change of the objects in the world by gaze shifting and verging the robot head. It should be pointed out that both gaze shifts and vergence movements are *rotational motions* with respect to certain axes of the robot head.

The reasoning system provides information as to which region should be picked as the visual target. A reference point should be retrieved from such region so that the motion parameters for the robot head can be computed. The *centroid* of the target region is possibly the best reference point, and is also relatively easy to calculate if the labelling procedure of the reasoning system can collect some additional statistical data.

Computing the motion parameters for the robot head is a process of converting a flow value to a gaze shifting command, and converting a disparity value to a vergence movement command. The conversion an inverse process to the computation of the expected background flow in the perception system. The special mapping table created during the initialization procedures can be used here for computing how much the head should pan and tilt. Knowing the depth of the region of interest, and how much its centroid is offseted from the center of the image, it is quite easy to work out the pan and tilt parameters using the mapping table. Finding out how much to verge requires the knowledge of the focal length of the cameras. A simple geometric equation, based on the configuration of the robot head, can be used to compute the vergence angle adjustment which will drive both cameras to point at the centroid of the visual target. More details on computing the motion parameters will be presented in next chapter.

Although the reasoning and action systems are established to be parallel processes, they are perhaps not intended to be concurrent or independent. The reasoning system should wait until the action system finishes moving the robot head before working on the next set of accumulated optical flow data (see figure 3.6). This way, the accumulation in the perception system can take in account how much apparent flow has been caused by the action system, and allows the cancellation process in the reasoning system to work with better approximations.

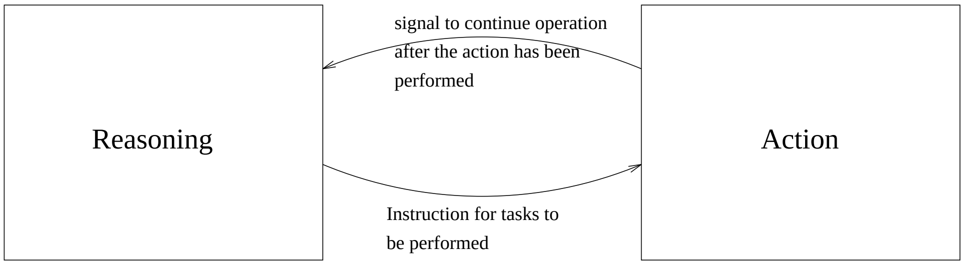

Figure 3.6: Communication between the Reasoning and Action Systems

The movement carried out by the robot head should be slow and smooth, so that the correlation system can generate less mismatches due to the fact that only a limited range of displacement measures is being focussed on.

By following this simple set of control strategies, we should be able to allow our passive motion tracking system to center the image of the object of interest in real-time.

3.3 Discussion of a Prediction System

Prediction is the key to stability of interacting closed-loop control systems with time delays [Brown, 1989]. Time delays are usually caused by interactions among different control systems, processes of acquiring real data, and computations performed on such data. For any real-time system to be successful in simulating or exhibiting realistic behaviour found in biological systems, it is often necessary that the response time of such system be as minimal as possible.

Our proposed control scheme is aimed to function as a passive motion tracking system. It is certainly our hope that by using powerful computer equipment, the response time during each PRA loop can be minimized to an acceptable level of operation. However, in practice, such assumption usually leads to unexpected breakdowns in unforeseen circumstances. In order to keep up with the pace of the world on a regular basis, a prediction module can assist the operation of the motion tracking system by indicating what are to be expected in future states.

A prediction module, possibly composed of a Kalman filter as used in [Singh, 1991] or [Matthies *et al.*, 1988], can undoubtedly enhance performance by compensating to the time delays in computation and communication. Based on our smooth movement constraint on both the moving objects and the moving observer, we can deduce that velocity changes will also be slow and smooth. This observation can be used as a key to predicting how far the objects are expected to move away from their current positions while the reasoning system is analysing the perceived data. Under this framework, the prediction module outputs the estimated destinations of the moving objects based on their current velocities, and the velocities changes previously undergone respectively.

The application of prediction in motion tracking involves having a *mediator process* taken into consideration, for the visual target, both the estimated distance to be travelled by such object during time delays, and the actual change in locations of the object computed from the perceived data. Such process will then be responsible to compute a *corrective measure* to drive the robot head to track such target. This is obviously a vast improvement to a passive system.

Prediction allows a motion tracking system to look ahead to the future states, and also allows the robot head to move ahead to the destination during time delays. That is, the robot head will still be moving along with the object while data is being analysed. This control scheme however contradicts somehow with the concept of being passive and responsive. There are certainly many open questions, which require in-depth investigation, concerning the feasibility and methodology of applying such a prediction module to our proposed control scheme of motion tracking. As a result, prediction is not included as part of our current study of designing and implementing a gaze control system. However, it is clearly one area which needs to be explored in future studies.

Chapter 4

Implementation

4.1 Overview

This chapter describes the implementation of a real-time motion tracking system based on the control theories presented in the previous chapter. The general hardware setup for the LCI robot head will be described. Different programs running on the Datacube hardware and on the Transputer network will also be discussed in detail in this chapter.

The evaluation of our motion tracking system, the analyses and results of different experiments, and some suggested improvements will be presented in the next chapter of this thesis.

4.2 Hardware Configuration for the LCI Robot Head

This section describes the hardware configuration of the binocular camera head system installed in the Laboratory for Computational Intelligence (LCI) at UBC. The system consists of a robot head, or otherwise known as the Eye-Head, and a special purpose image processing system together with a multicomputer system, both being connected to a host computer (see figure 4.1).

Correlation is done on a Datacube MaxVideo-200 system, which has a pipelined architecture. Digitizing of images of size 512 by 480 on the MaxVideo system can be performed at video rate, i.e., the rate of 30 frames a second. Processing then proceeds at 60 frames per second. Processing smaller images takes correspondingly less time. The Datacube system is attached to the VME bus of the host workstation.

A network of T-800 Transputers running at 25MHz is also connected to the VME bus of the host, a Sun SPARCstation 2 with 32 MB RAM. Each Transputer processor has at least 2 MB RAM attached, with some special nodes, such as the frame grabber, containing more memory. Each Transputer can be connected to another Transputer through one of its four links by a custom-built Crossbar switch. The bi-directional link is capable of transferring data at a speed of 20 megabits per second, which is roughly twice the Ethernet data transfer rate. The Transputer network can share data with the Datacube system via a MaxTran node, a special node consisting of a Transputer, a frame buffer, and a link to the MaxVideo board. More details on the Datacube hardware and the Transputer network used in LCI can be found in [Little *et al.*, 1991].

The LCI Robot Head contains a pair of Sony XC-77RR CCD monochrome cameras mounted on a motorized platform. The focal length of the camera is 8.5 mm. Two Cosmicar lenses, with manual focus and aperture control, are used. The cameras are mounted on the platform approximately 20 cm apart. The output of the cameras can be grabbed by either the Datacube system or a frame grabbing node in the Transputer network, allowing more flexibility in designing the image processing routines.

The robot head (see figure 4.2) made by Zebra Robotics can be electronically con-

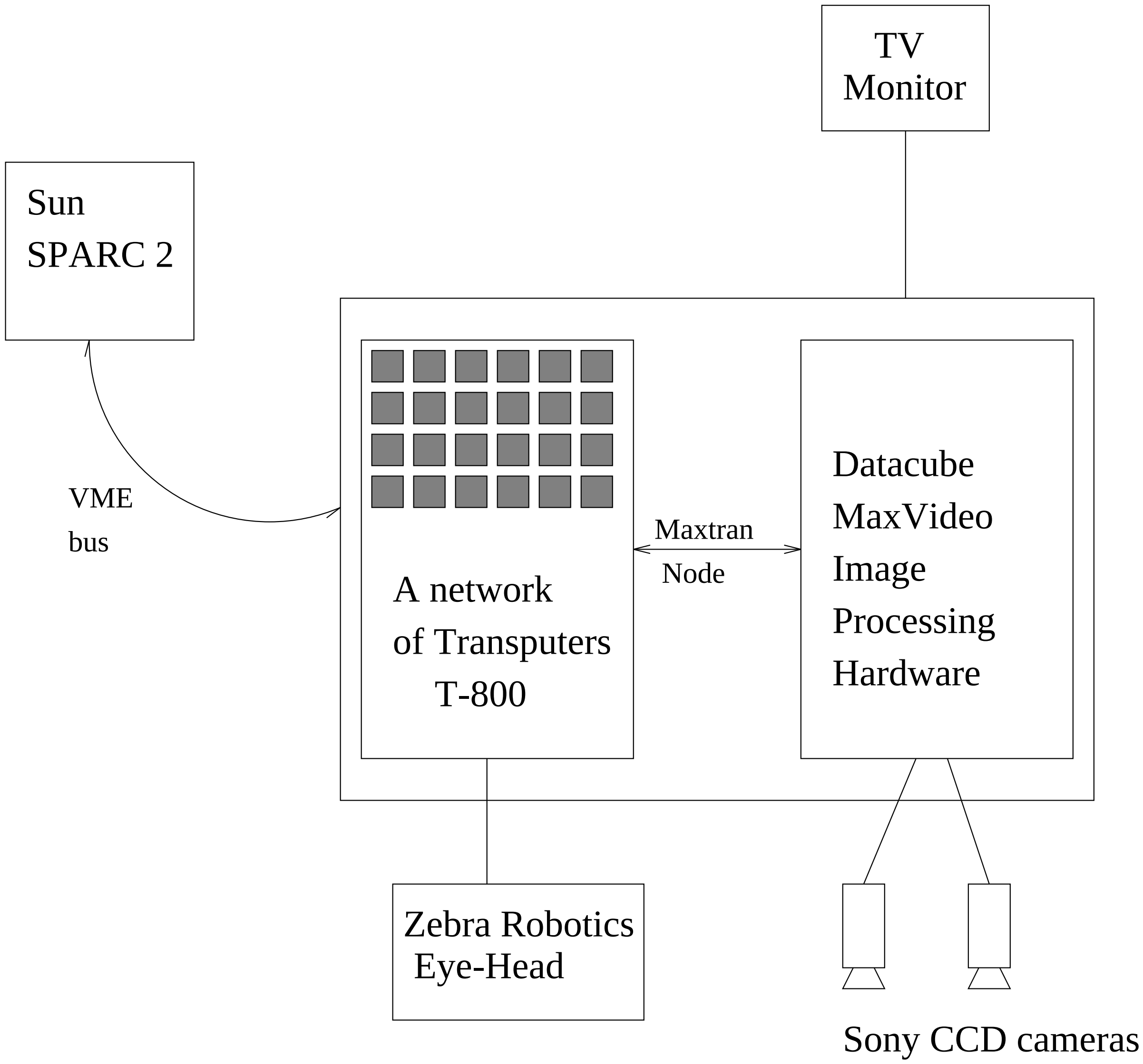

Figure 4.1: Hardware Configuration of the LCI Robot Head System

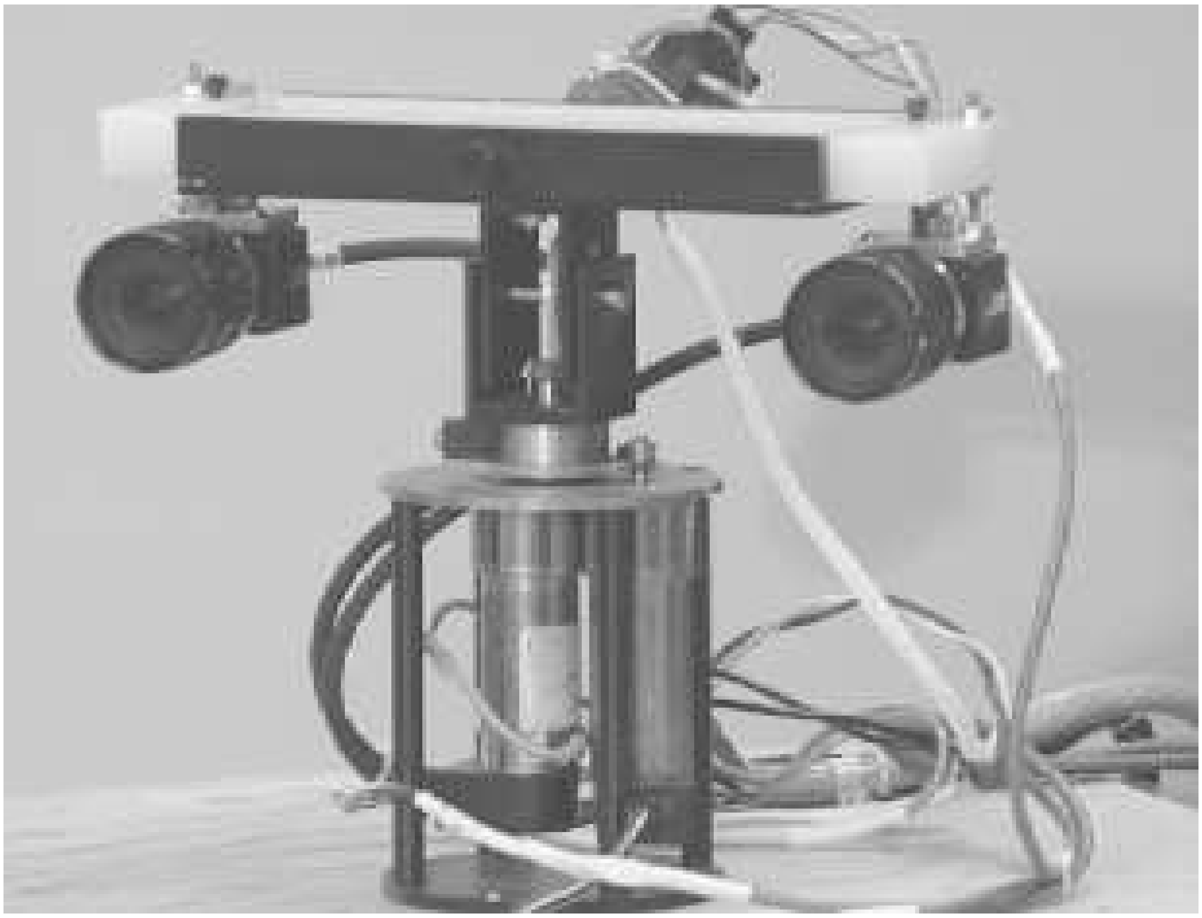

Figure 4.2: The LCI Robot Head

trolled to pan, tilt, and verge with respect to some axes (refer to figure 4.3). The pan axis has a 4:1 spur gear reduction ratio, while the tilt axis has a 3:1 bevel gear reduction ratio. The pan and tilt axes are coupled, in that one revolution of the pan axis will cause a one-third revolution of the tilt axis. The verge axis is driven by a 20 threads per inch lead screw. Pittman motors are used, with each motor being connected to an HP encoder board. The motor amplifiers are driven by a digital-to-analog converter controlled by a Transputer in the network. The encoder boards are linked to a special Transputer node, and therefore the head motion can be controlled by software.

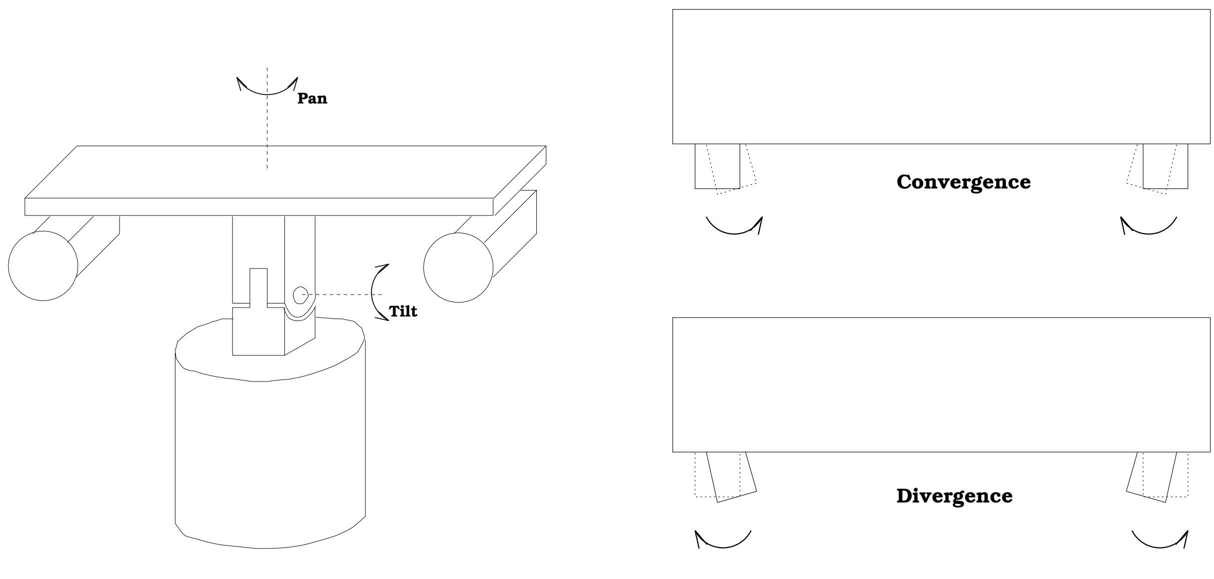

Figure 4.3: Various Motions of the Robot Head

The pan axis supports motion ranging from roughly -150 degrees to +150 degrees. The motion range for the tilt axis is -90 degrees to +65 degrees. The vergence angle of a binocular system is the angle between the optic axes of its cameras. The maximum vergence angle allowed is roughly 20 degrees. The velocities of the head motions can be controlled by adjusting parameters during the head initialization process. The maximum velocity of the head is approximately 120 degrees per second (dps) for panning, 150 dps for tilting, and 15 dps for verging.

4.3 Software Description

As suggested by our control theories, both the Datacube hardware and the Transputer network will be used in our implementation of the motion tracking system. The role of the datacube program is to compute optical flow and stereo disparity repeatedly, and to return the data in a continuous flow of images. These images are put into the frame buffers of the MaxTran node, allowing a Transputer process to access the data. Programs running on the Transputer network will be responsible for storing and analysing the displacement data, and for controlling the motion of the robot head.

Figure 4.4 shows the software design of the overall system. It also depicts the arrangement and relationships of the different software components implemented, along with a brief description of the data flow in the network of Transputer programs. A circle in the figure denotes a Transputer node loaded with a program with a specific name, and the number enclosed in the circle identifies the particular Transputer node used for the program. An arrow indicates the direction of the data flow from one Transputer node to another. Data is being transferred via message passing in the Trollius operating system running on the Transputers. The functionalities and purposes of each program written will be described in detail in the following sections.

4.3.1 The Datacube Program

A datacube program has been written by Stewart Kingdon, one of our LCI staff, to compute optical flow and stereo disparity using the Datacube MaxVideo hardware. The input images are grabbed from the pair of stereo cameras. The images are taken at 30 frames per second; we only use one field of the frame. The images are 512x240, our Sony cameras average successive odd and even pairs so that we do not have missing data

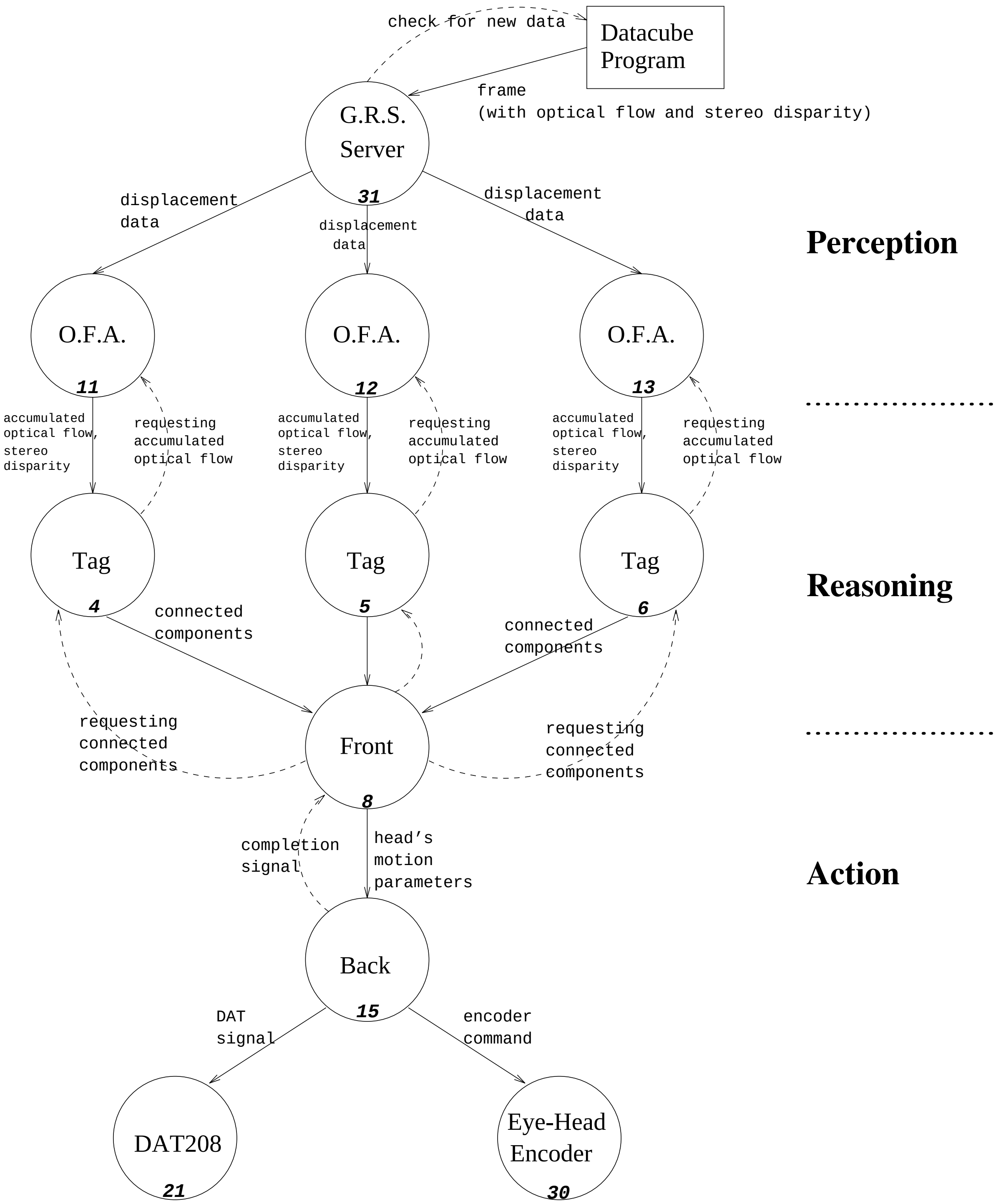

Figure 4.4: Software Components and Data Flow

in the vertical direction. We smooth with a Gaussian before subsampling to 128x120 [Little *et al.*, 1991]. An output image of size 128x512 is returned with the following four subframes, which are in order: optical flow, stereo disparity, edges (zero-crossings of the Laplacian of Gaussian), and Laplacian of Gaussian images (see figure 4.5). Each of these subframes has a default size of 128x128, which can be scaled to a lower resolution of 64x64 if needed. The output can be displayed on a TV monitor, in which different colors are chosen to represent different optical flow values, and stereo disparities.

Optical Flow	Stereo Disparity	Edges	$\nabla^2 G$

Figure 4.5: The Four Subframes in the Output Image of the Datacube Program

The range of motion to be considered is specified in an input data file to the datacube program. The common range of motion used is [-2, +2] for vertical flow and [-3, +3] for horizontal flow, and the range for stereo disparity to be considered is usually [-13, +13]. The input data file also contains some *encoders* used by the datacube program for releasing the displacement data. Encoders are used so that only one optical flow image can be returned with both horizontal and vertical flow at every pixel of the image. Any program that requires access to optical flow or stereo disparity can read the particular data file used by the datacube program, and *decode* the encoded result. The data file also contains information of the color used for each optical flow or stereo disparity data value. The datacube program is capable of returning about 10 128x512 output images per second with the input range of motion and disparity mentioned above.

Simple sum of absolute values of differences (SAD) correlation technique is used in measuring optical flow and stereo disparity. A 7x7 correlation window is used in the computation. Only the correlation range specified in the input data file will be considered. If there is no good match because of insufficient information, then no correlation result at that particular point will be returned.

4.3.2 The Transputer Programs

Trollius 2.1 operating system is used in our implementation of the motion tracking system described as follows. Trollius is a programming environment designed for distributed memory multicomputers [The Ohio State University, 1991]. The environment includes an operating system, a user interface, a C compiler, and some libraries for C. A boot schema, which specifies the identifiers and types of nodes, and their interconnections, is required to set up the topology of the Transputer network. The current interconnection of any two Transputers follow the regulation that link 0 connects to link 1, and link 2 connects to link 3. Programs written in C can be loaded onto the nodes after the booting process has completed.

Various programs (refer to figure 4.4) running in parallel have been written by the author of this thesis to model our PRA control theories. The displacement data is first handled by a **G.R.S.** program, which is responsible for grabbing the displacement data returned by the datacube program, rearranging the data if necessary (e.g., resample), and sending them to other processes. The optical flow accumulation (**O.F.A.**) program will keep the optical flow in some pre-allocated storages, and send out the accumulated data when requested by a process in the reasoning system. The **Tag** program will be responsible to make such a request and to segment the accumulated optical flow into different connected components. The **Front** program acts as the commander of the reasoning system to lead the data flow and to request that the accumulated optical flow be segmented. A decision process in the front program will select the visual target from the connected components, and allows another procedure to compute the motion parameters for the robot head. Such motion parameters, or motor commands, will be sent to the **Back** program, which in turns instructs the robot head to move by sending signals to the encoder boards, and to the digital-to-analog board. A special **Stop** program has been written to send a *quit* signal to both the front and G.R.S. programs, requesting that the ongoing operations be terminated.

4.3.2.1 The G.R.S. Server

The Grab-Rearrange-Send server acts as the *active monitor* of the perception system responsible for dealing with the displacement data being pumped out from the datacube program. This server is aimed to run on the MaxTran node so that data can be accessed directly from the frame buffers. The data received in the frame buffer will be the image containing the four subframes as described in a previous section.

The grabbing process will constantly check for new data to arrive, and immediately rearrange the new data if necessary. The rearrangement includes the possibility of resampling the data to a lower resolution in order to speed up the response time. This is certainly an optional service of the G.R.S. server, and should not be used unless it is absolutely necessary. It should be noted that resampling of data is preferred to be performed by the datacube program before the correlation process, so that displacements show the measurements with respect to the image size after resampling, rather than before resampling. Such ordering of events is crucial in later processes in which offseting errors might seriously affect any result, if resampling is done to the images after displacements have been measured with respect to the higher resolution image size.

The final step is to send displacement data to the optical flow accumulators. Data is sent between two Transputer nodes via the message passing mechanism in the Trollius operating system.

The G.R.S. server runs as a continuous loop, actively repeating the grabbing, rearranging, and sending operations without requiring any instruction from any parent process. It is, however, alert to a *quit* signal indicating when such operations should stop. The server is intended to run at a rate no slower than the displacement data is being returned, and therefore, the workload of the server is assigned to be as minimal as possible. It is necessary that the grabbing process be able to attend to all images. However, it is anticipated that a single accumulation process may not be able to handle the amount of data due to the time delays of processing, and as a result, a number of optical flow accumulators, each responsible for processing a portion of the image, should be employed to speed up the operations.

4.3.2.2 The Optical Flow Accumulator

In our current implementation, we elect to use *three* optical flow accumulators running on three separate Transputers at the same time. On our previous attempts to accumulate optical flow with only a single accumulator, images were lost because the accumulator could not execute fast enough to manage all data.

Each of the three accumulators in the current system is responsible for processing only one-third of the original optical flow image. The accumulation process is a separate looping process actively working to manage the incoming data. The so-called foreground process of the program is the one waiting for signals expressing the need for accumulated optical flow. Two buffers are required for accumulation because one would not want to block the accumulation process when there is a need to send data. As a consequence, the two buffers will alternate their roles from time to time, one responsible to store accumulated optical flow data, and the other being used by the process attempting to send data over to a recipient process on another Transputer.

Optical flow accumulation, as we described before, is merely simple vector addition. The encoded optical flow data contained in the output of the datacube program will be decoded into separate horizontal and vertical flow values, before they are being added to the existing flow data. The actual accumulation works by first looking at the flow values at the current pixel location, and then retracting the flow values at the position where the current flow values direct to, and finally adding these retrieved flow values to the current flow values.

For instance, suppose we are trying to update the flow value of pixel (x, y). Assume that

$$h\_flow_{0,i}(x, y) = h$$

$$v\_flow_{0,i}(x, y) = v$$

at time i. Then,

$$h\_flow_{0,i+1}(x, y) = h + h\_flow_{i,i+1}(x+h, y+v)$$

$$v\_flow_{0,i+1}(x, y) = v + v\_flow_{i,i+1}(x+h, y+v)$$

where $(i, i+1)$ denotes the new optical flow image.

The accumulation program is built so that an image is completely processed before the next one is being attended to. This means that the image sending process of the G.R.S. server will be temporarily blocked if the accumulation process is not really for the data. The idea of queuing up images on this receiving end will not work, as once the accumulation process falls behind the pace, it will not be able to catch up with the real-data. Therefore, downsizing the images will be a more logical choice in this real-time system, when time constraint is a major factor in performance.

Another common route to speed up response time, or to avoid image loss, is to add more accumulators. This is the perfect solution in terms of utilizing resources and maximizing the amount of data to be processed. Unfortunately, we are unable plug in more optical flow accumulators to the current structure of the Transputer network, since there are only 4 links connected to a single Transputer. It is possible to add in a layer of optical flow transmitters, and therefore, allows more accumulators to work at the same time. However, the time used for transferring data will increase significantly, and it will not likely help to speed up the operations by a significant factor.

4.3.2.3 The Tag Program

The tag program is responsible for segmenting the accumulated optical flow into different connected components. The segmentation process is activated by the need of the front program for picking a visual target. The number of tag programs, or workers, is equivalent to the number of optical flow accumulators, in which each tag worker is paired up with a particular accumulator at all time. The front program will broadcast a request signal to all tag workers for connected components. Upon receipt of such signal, each tag worker will request the accumulated optical flow be sent over by its respective coupled accumulator, and then transfer the accumulated data to the segmentation process.

Based on our control theories, a cancellation process is used to reduce the unstable effect of the background optical flow caused by ego-motion, before the segmentation process can work on partitioning the flow field. Such cancellation process is integrated into the segmentation process in our implementation, in the way that whenever the flow values of a pixel are retreived from the flow field, the expected background flow computed for such pixel will be subtracted from the flow values, resulting in a revised pair of horizontal and vertical accumulated optical flow.

The expected background flow values for each pixel is, in theory, calculated by multiplying the ego-motion parameters with the mapping values corresponding to the depth of such pixel. A special mapping table, indexed by stereo disparity, is recommended for use with the cancellation procedure. However, in our current implementation of the motion tracking system, *average mapping* values, one for vertical motion and another for horizontal, are used instead of a special table. This approach is adapted because of the difficulty of establishing a reference point for each depth, without knowing for sure that such a point will exist when the cameras are pointing at a random direction during initialization. It is also our belief that since cancellation will not completely zero out the background optical flow due to round off errors and correlation errors, using average numbers are as good in approximation and creation of "pop-out" effects as using a complete table, whose entries are also subject to contain errors due to the use of correlation matching. As a result, the initialization process will move the head in a certain degree on each axis, and compute how much optical flow has been incurred, assuming that the whole scene is stationary. Such procedure will be repeated a number of times, using different degrees of motion, before the average mapping values are derived.

We implement the segmentation process using a labelling algorithm. The *magnitude* and *direction* of a vector are used as basic features for defining connectedness in our algorithm. Two vectors will be loosely defined to belong to the same region, or match, if the differences of their magnitudes and directions are less than some pre-defined thresholds. The labelling algorithm works as follows:

> For each vector v at position (x, y), its features are first compared with the features of the region containing vector v_1 at position $(x, y-1)$. If both features match, then vector v should belong to such connected component, and share the same tag (or label) with vector v_1. A checking procedure is required to find out if the region containing vector v_2 at position $(x-1, y)$ has

matching features with the region containing vector v. If it is the case that they match but do not share a common tag, i.e., the two regions have disjoint lists of tags, then a merging process will be used to group the two regions together so that both regions are combined into one connected component.

If the features of v do not match with the features of the region containing vector v_1, then the features of the region containing vector v_2 will be compared with the features of vector v. If they match, then vector v belongs to such region and share the same tag with vector v_2. Otherwise, vector v belongs to a brand new region of its own, and will be assigned a new tag.

If either vector v_1 or v_2 does not exist because v is at the boundary line, then no comparison will be performed. The algorithm continues until all vectors in the flow field are grouped into some connected components. It should be obvious that a connected component can consist of numerous tags, but each tag can only belong to one unique region.

Consider the optical flow field in figure 4.6 as an example. Assume that the labelling process scans the field from left to right and from top to bottom. The vector at position (4,3) starts a new connected component with tag number 1. Vectors (5,3) and (6,3) belong to the same region and are assigned the same tag number since they have similar features with vector (4,3). The vector at position (9,3) creates a new region with a different tag number as there is no flow value at positions (9,2) and (8,3). This particular region will be merged to the initial connected component after the vector at position (9,4) has been analysed. This is due to the fact that vectors (9,3), (9,4), and (8,3) have similar features and should belong to the same region. As a result, the initial connected component expands and its tag list also grows. Similar procedures are applied to vectors (4,5), (3,6), and (2,8). The vector at position (9,8) begins a new region since its features do not match with those already examined. In this example, the labelling process finishes with two separate connected components, one with 39 vectors and containing five tags, and the other with just 5 vectors and using only one tag.

The definition of connectedness should also include stereo disparity as a constraint

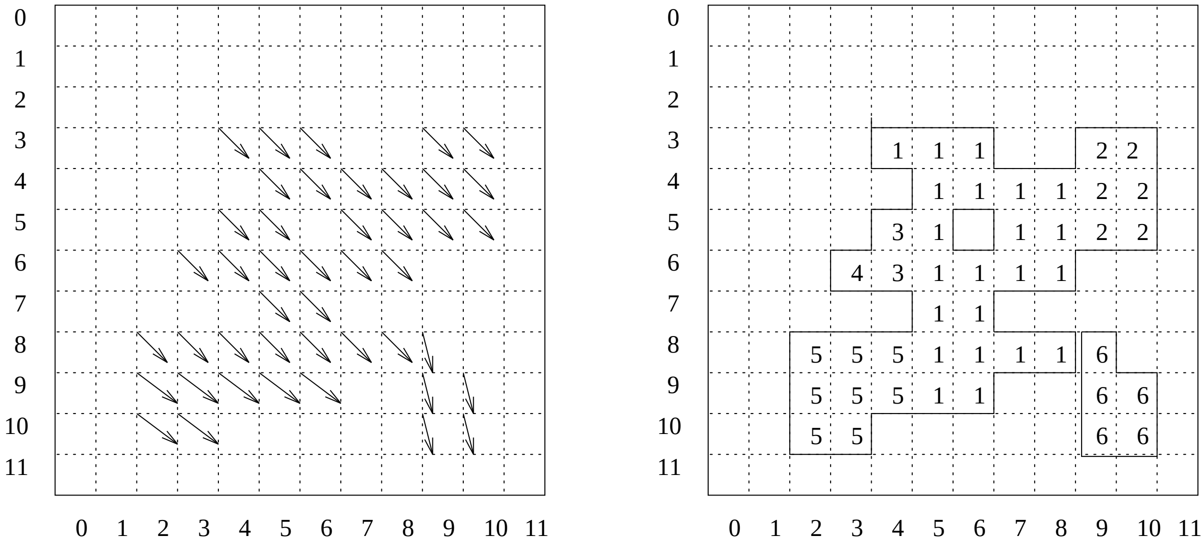

Figure 4.6: An Example Optical Flow Field and the Result Returned by the Labelling Algorithm

if we are to create a motion tracking system which works in three dimensions, i.e., to pan, tilt, and verge at the same time. It is still an open question, however, as to how the depth constraint fits into the algorithm, as we try to avoid running into situations where the flow field is *overly divided* into different connected components. This situation arises if the constraints used are too firm, and that we may end up losing the target quite easily. Similarly, if the constraints are too flexible or loose, the whole flow field may not be partitioned at all, and thus the segmentation process becomes useless. It should be obvious that the effects of any constraint are based on the values of the *thresholds* used, and it is our hope that subsequent experimentations can provide some useful and reliable thresholds for the system.

During the segmentation process, a number of statistical data will be collected for later computation. Particularly, the number of pixels in each region, and those that correspond only to horizontal motion, or vertical motion, will be keep tracked of. The minimum and maximum boundary values of each component are useful in future processes such as merging and determining the rough positions of the region. The various sums of vertical optical flow, horizontal optical flow, and stereo disparity within each connected component are computed. A stereo disparity histogram is constructed to determine the depth of a region in a later procedure. Other useful data for computing the centroid of a region are also collected in this segmentation process.

4.3.2.4 The Front Program

This Transputer program can be regarded as the main controller of the activities carried out by the reasoning system. The tag program plays a passive role in the operations of the system by waiting for signals to start up the segmentation process. This front program has an active loop requesting that the connected components be sent over by the tag workers, and deciding which of those components should be used as the target to be tracked.

Upon receipt of the connected components, a selection procedure will decide which region is the *right* or *best* one to follow. As discussed already in the previous chapter, this is a rather ambiguous task which really depends on how the motion tracking system is defined to behave. Based on the observation that if an object is being followed in three-dimensional space, its disparity should be close to zero, and its centroid should be fairly close to the center of the image if the motion of such object is relatively slow. Using such ideas as the selection criteria will at least ensure that once the target is being followed, the system will likely continue to track its motion. However, it does not address the initial objective of having our motion tracking system paying attention to multiple moving objects. In order to shift attention to other moving objects, there have to be some conditions, such as it is more important to pay attention to the closest object or the largest moving region is more interesting, to override the zero-disparity criteria. However, doing so may cause the robot head to move aimlessly on noisy images. Different selection routines have been written to pick out a connected component, using the data collected by the segmentation process. Only testings can demonstrate which method is the better behaving one.

The front program is also responsible to compute the centroid of the chosen target. The centroid (x_c, y_c) can be calculated by the following equations:

$$x_c = \frac{\sum x_i}{n}$$

$$y_c = \frac{\sum y_i}{n}$$

where x_i and y_i are x and y positions of a point belonging to the region, and n is the number of points in the region.

The centroid will be taken as the representative point of the region to be tracked when computing the motion parameters of the robot head. The motion parameters, once computed, will be passed on to the back program for action.

4.3.2.5 The Back Program

The only responsibility of the back program is to move the robot head, according to the motion parameters received. Signals will be sent over to the digital-to-analog converter board and the encoder boards to drive the various motors. The low-level controlling routines, and some inverse kinematics functions have been provided by Rod Barman, one of our friendly LCI staff. The details of such procedures are beyond the scope of this thesis.

It should be re-iterated that the head motion must be slow and smooth in order to allow our control theories and correlation matching technique to function properly. The velocity and acceleration of the robot head can be adjusted to a suitable level during the head initialization process. Once the head has moved to its destination, a completion signal will be sent back to the front program, indicating that processing can be continued for the next set of data.

4.4 Computing the Motion Parameters of the Robot Head

4.4.1 Panning and Tilting

Once the centroid of the visual target has been computed, the pan and tilt motion parameters of the robot head can easily be determined. The goal of tracking is to keep the region of interest at the center of the stereo images. Finding out how much the centroid has moved away from the center of an image is an easy task. Such differences can be simply converted, using the average mapping values, to the degrees of motion that the robot head should undertake to follow the moving region. However, the centroid of the region identifies the location of the object *before* the motion, rather than after the motion. This is due to the fact that our optical flow accumulation process adds new flow values to the origin of the flow. That is, the coordinate system of the accumulated optical flow image is the one used in the first image frame of the motion sequence, and the optical flow values point to the destinations of the respective points at the end of the sequence. As a result, the pan motion parameter should also take into consideration the average horizontal optical flow of the region, and the tilt motion parameter should be computed also using the average vertical flow value of the region. The expected background optical flow previously subtracted from the flow field should be added back to the average flow values, so that the pan and tilt motion parameters reveal the *true* displacements of the target object with respect to the robot head after its motion rather than before its motion.

The formulae used are:

$$\delta_{pan} = \frac{x_c - \frac{w}{2} + h_{average} + h_{background}}{map_{horizontal}}$$

$$\delta_{tilt} = \frac{y_c - \frac{w}{2} + v_{average} + v_{background}}{map_{vertical}}$$

where δ_{pan} is the pan motion parameter, δ_{tilt} is the tilt motion parameter, and w is the size of the image, assuming that the width and the height of the image is the same.

The resulting motion parameters, which are measured in radians, direct the robot head to move to a new location, and the cameras will still be *looking* at the object being tracked, if the total response time of the motion tracking system is compatible with the speed of the moving object.

4.4.2 Verging

Vergence movement is a coupled motion of the two cameras wherein the cameras rotate in opposite directions. Determining the verge motion parameter requires understanding of the geometry of using stereo cameras, and their image planes. Referring to figure 4.7, the two stereo cameras can be verged inward or outward. The two image planes are therefore *not* necessarily coplanar, but the optical axes of the cameras should still lie in the same plane.

In the diagrams, *cv* denotes the *current verge* angle with both cameras pointing at

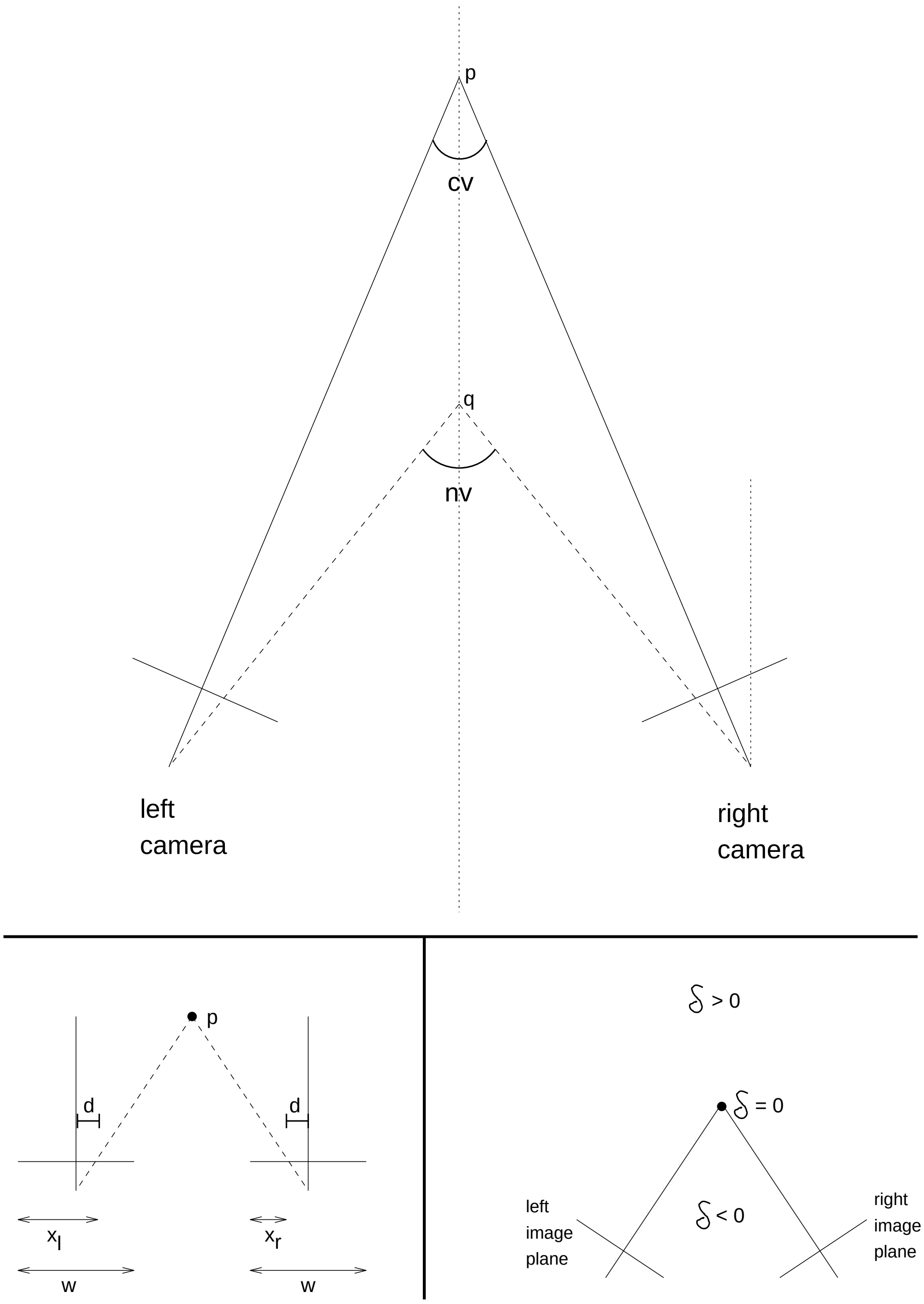

Figure 4.7: The Geometry of Using Stereo Cameras

point p, and nv denotes the *new verge* angle if the cameras are to direct attention to point q. The disparity δ of an image point p is defined to be:

$$\delta = x_r - x_l \tag{4.1}$$

where x_r is the location of the pixel found on the right image plane, and x_l is the location with respect to the left image plane.

Let us assume that the motion tracking system can react fast enough to follow the moving object, so that the object always appears close to the center between the stereo cameras.

Based on the *symmetrical* movement of our stereo cameras on the motorized platform, we can assume that the centroid, being close to the center between the two cameras, is symmetrically displaced from the center of the image plane, so

$$d = \frac{1}{2}w - x_r = x_l - \frac{1}{2}w$$

as can be seen from the diagrams, and therefore

$$x_l + x_r = w \tag{4.2}$$

where w is the size of horizontal dimension of the image plane.

The point that the two optical axes intersect is usually referred to as the *vergence point*, which has zero disparity. For points that are closer to the cameras than the vergence point, their disparities are always less than zero, and for any point which is farther away from the cameras than the vergence point, the disparity is always greater than zero.

Figure 4.8 expands the view of the right image plane as introduced in figure 4.7.

When the cameras verge, the image planes will follow. By simple geometry, we are able to identify the values of the angles as labelled. The angle δ_{verge} represents the degree

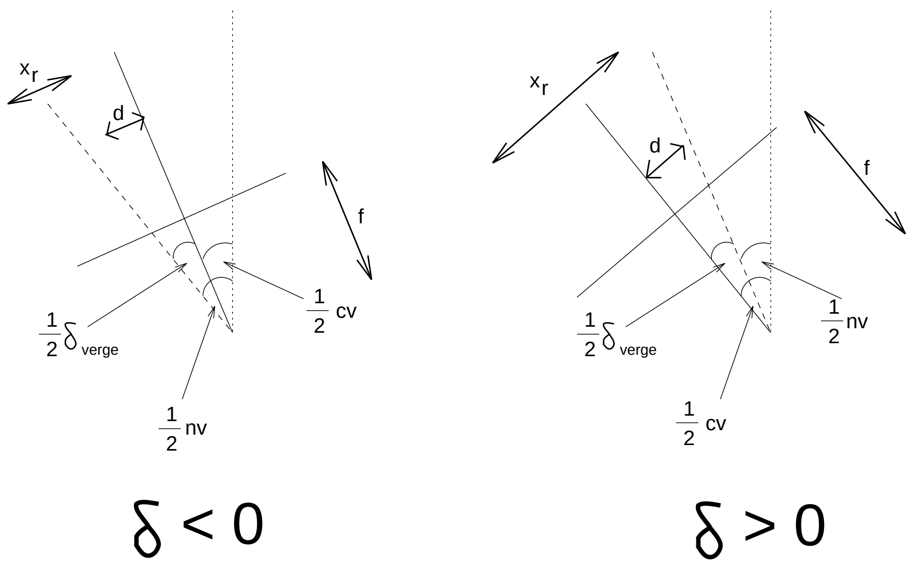

Figure 4.8: Determination of the δ_{verge} angle

of vergence movement required in order to change the total vergence angle from cv to nv. From the diagram, we can observe that

$$\frac{1}{2}\delta_{verge} = \frac{1}{2}nv - \frac{1}{2}cv \qquad \text{if } \delta < 0$$

$$\frac{1}{2}\delta_{verge} = \frac{1}{2}cv - \frac{1}{2}nv \qquad \text{if } \delta > 0$$

and f is the focal length of the cameras.

The goal here is to compute δ_{verge} based on the disparity on the region to be tracked. We know that

$$\tan\left(\frac{1}{2}\delta_{verge}\right) = \frac{d}{f}$$

where

$$d = \frac{1}{2}w - x_r \qquad \text{if } \delta < 0$$

$$d = x_r - \frac{1}{2}w \qquad \text{if } \delta > 0$$

Given that f is known, we need to retrieve x_r from disparity δ. From equation 4.1, we know that

$$x_r = \delta + x_l$$

and from equation 4.2, we know that

$$x_l = w - x_r$$

Therefore, we can derive that

$$x_r = \frac{\delta + w}{2}$$

and

$$\tan\left(\frac{1}{2}\delta_{verge}\right) = \frac{\frac{1}{2}w - \frac{\delta+w}{2}}{f} \qquad \text{if } \delta < 0$$

$$\tan\left(\frac{1}{2}\delta_{verge}\right) = \frac{\frac{\delta+w}{2} - \frac{1}{2}w}{f} \qquad \text{if } \delta > 0$$

Finally,

$$\delta_{verge} = 2 \times \arctan\left(\frac{\frac{-1}{2}\delta}{f}\right) \qquad \text{if } \delta < 0$$

$$\delta_{verge} = 2 \times \arctan\left(\frac{\frac{1}{2}\delta}{f}\right) \qquad \text{if } \delta > 0$$

The direction of δ_{verge} is based on the the sign of δ. As a result, we end up with only one equation:

$$\delta_{verge} = 2 \times \arctan\left(\frac{\frac{1}{2}\delta}{f}\right)$$

The focal length f should be measured in unit of pixels. A CCD camera calibration process [Beyer, 1992] is technically required to perform such measurement; similar work is currently in progress. The value of f that we use in our experiments is obtained from a trial-and-error method.

We should re-iterate that our derivation of δ_{verge} relies heavily on the assumption that the reference point we are focussing on lies on the center plane between the stereo cameras. This is a fair assumption if the motion tracking system can center the region of interest as quickly as possible, and thus allows the derivation to come up with a reliable estimate of how much vergence movement is required.

Chapter 5

Evaluation and Discussion

This chapter presents the evaluation of our motion tracking system implemented as described in the previous chapter. Several experiments have been performed to demonstrate what can be done and what cannot be done by our motion tracking system. We will find out as to what extent our system is able to operate satisfactorily, and will discuss the problems which we have encountered in our experiments.

We will attempt to find solutions to the questions being brought up in previous discussions, as well as some issues which have to be addressed in future research. We will briefly compare our motion tracking system with other different types of systems implemented elsewhere. We will suggest and describe some ideas to improve our current system in the next chapter.

5.1 Evaluation and Performance of our Motion Tracking System

The performance and usefulness of a real-time motion tracking system can be measured in terms of the following parameters:

- the amount and size of data that can be handled
- the update rate, and response time
- robustness, i.e., accuracy and stability
- the computational resource required
- the maximum speed of a moving object that can be followed
- the maximum velocity of the robot head that is allowed
- the amount of data that can be produced, perhaps for use by some other programs at the same time

The datacube program works on gray-scaled images input from the stereo cameras, with the ranges for correlation specified in an input data file. In the experiments that we have carried out, we generally use the correlation range [-3, +3] for horizontal motion, [-2, +2] for vertical motion, and [-13, +13] for stereo. Each subframe of the 128x512 output image is selected to be of the default size 128x128. The datacube program reports an output rate of approximately 10 new images per second based on this configuration. It should be noted that the datacube output is being updated at video rate, i.e., 30 Hz, but new data is available at the rate of 10 Hz. Experimentation reveals that our Transputer processes cannot perform their duties in real-time with the processing of the 128x128 optical flow images. It is therefore necessary that these output images be scaled down to a lower resolution. If rescaling of the optical flow and stereo subframes to the size of 64x64 is being performed by the frame grabbing Transputer process, then the active monitoring process cannot execute at a rate compatible with the datacube program's output rate, and as a result, the optical flow accumulators can only pay attention to about 95% of the output data.

Allowing the datacube program to rescale the output subframes to the size 64x64 will also get us only 10 new images per second, for the reason that the output image will still remain to be of size 128x512 but each of the four subframes is of size 64x64. Even though we are unable to speed up the output rate, rescaling being done on the Datacube hardware is still the favorable option because the offseting errors of resampling, such as round-off errors, after correlation will be eliminated. In addition, it appears that less noise is being generated in the lower resolution output. However, since the size of the input gray scaled images will remain unchanged, smaller image size for correlation means that smaller motion will not be noticed as often. For example, if the input horizontal image size is 512, then 4 pixels of horizontal movement will translate to 1 pixel of displacement in the 128 pixels wide output image, but 8 pixels of horizontal movement is required for the 64 pixels wide output image to record a pixel of displacement. Nevertheless, since there is less work to be done by the frame grabbing Transputer process, the active monitoring process and the optical flow accumulators can consequently attend to all output images returned at the rate of 10 Hz.

The various response time of the different Transputer programs usually depend on the nature and the goals of the experiments, and will be reported along with the results of the different experiments explained below.

We assume that the speed of any moving object and the robot head is slow and smooth when establishing our control theories. With the correlation ranges for motion used as mentioned above, our correlation process can produce reliable matches for an object moving with a speed at most 1.2 metres per second approximately, if the object is 2 metres away from the cameras. Moving objects that are farther away from the cameras can be supported at a slightly higher velocity. Objects that are moving faster than the speed limit cause unreliable correlation output, as can be instantly observed from the different color patches appearing on the TV monitor. Currently, we allow our robot head to pan or tilt with a maximum speed of roughly 15 degrees per second, so that the background velocities generated are always within the correlation ranges of the datacube program. These upper bounds for velocities are in general determined using the trial-and-error method.

5.1.1 Experiments, and What Can Be Done

Through experiments, we can show what our current system is able to achieve, study the advantages and disadvantages of our approach to motion tracking, and motivate new ideas for future improvements.

5.1.1.1 Detection of Moving Objects when Robot Head is Not Moving

Before we allow our robot head to track, we first have to confirm that the correlation system is returning usable data, the perception system is working properly in accumulating optical flow, and the reasoning system is able to segment the flow field and pick out the target. Therefore, our first and the easiest experiment is to simply detect the locations of a moving object in view over time without following it. This way, we can analyse the output of the reasoning system without being distracted by the cameras' motion.

The experiment is conducted with a person walking into the view of the cameras from the left side, and slowly moving to the right, and then walking back to the left and exit from view. It should be noted that for the purpose of easy verification of results, only one moving object is present in the image sequence.

The datacube program outputs the displacement data at a rate of 10 Hz. The active monitoring process is able to notice all output images, and the optical flow accumulators can attend to all the 64x64 optical flow images returned.

Not surprisingly, the tag programs can correctly segment the moving region from the stationary background, and the front program correctly reports that an object can be *seen* moving from the middle left to the right of the image and then back to the top left by monitoring the centroid and the boundary of the moving blob (see figure 5.1). In this experiment, we pick the largest motion blob as our visual target, and it appears to be working very nicely with slow moving objects. Each PRA iteration takes an average of 1.38 seconds to complete.

This experiment gives us the confidence that our perception system is functioning properly, and that we are able to detect and identify the locations of a moving object, although there is no background optical flow generated by the stationary cameras.

5.1.1.2 The Vergence Only Experiment

The second experiment that we have carried out is to control the vergence movement for fixation without panning or tilting the robot head. This experiment is conducted with a person walking towards and away from the stereo cameras, while keeping himself or herself close to the center of the images at all time. The input disparity range for

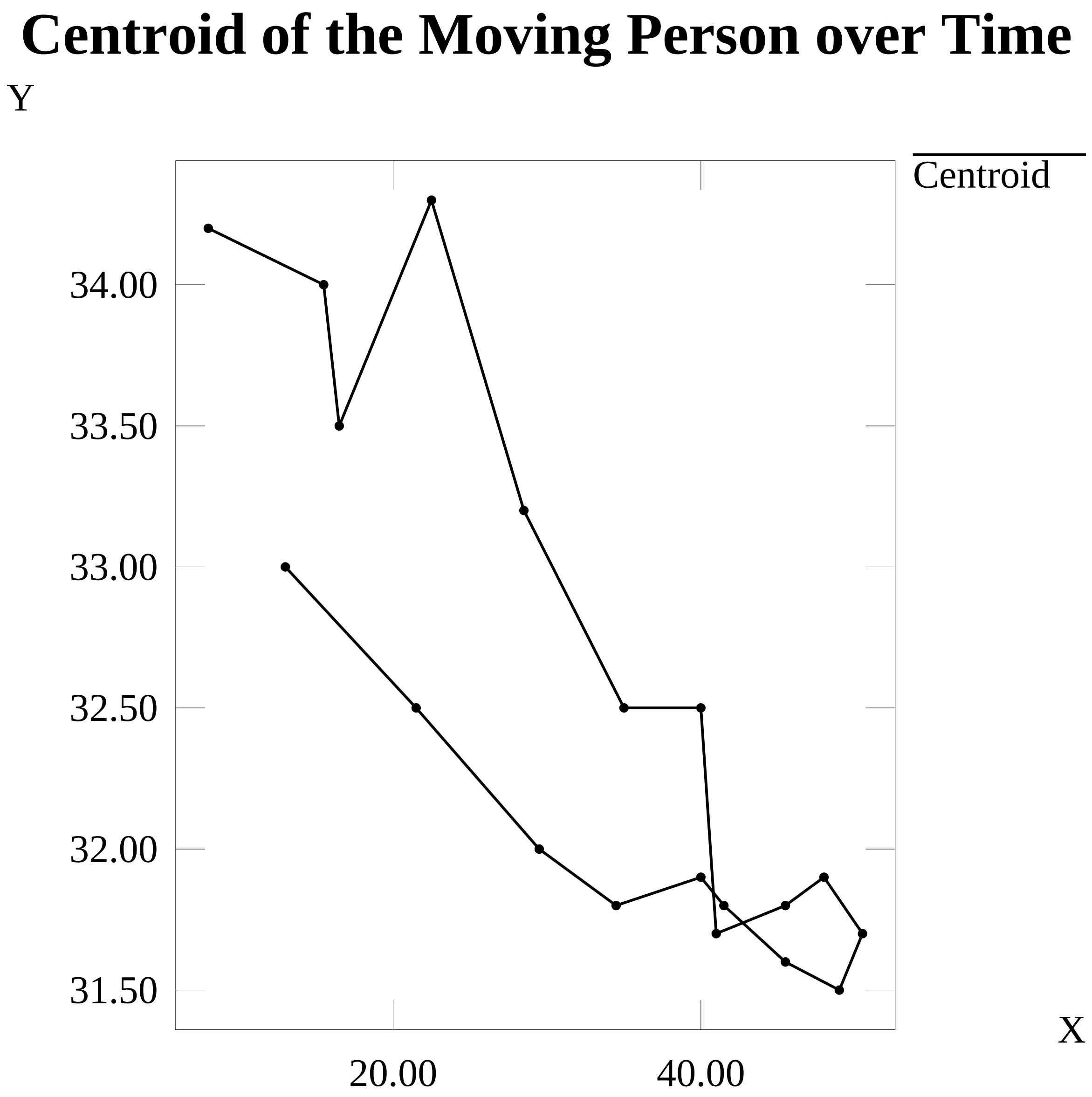

Figure 5.1: Results of the Motion Detection Experiment

correlation is [-13, +13], and the datacube program is pumping out images of 64x64 subframes 10 times a second. Only stereo data is required for this experiment, as we have already derived an equation for verging the cameras based on only disparity. It should be pointed out that the optical flow accumulators and the segmentation processes are not required for managing any data in this experiment. Stereo disparity images are transmitted directly from the frame grabbing server to the front program for processing. For the sake of simplicity, we are only analysing data in the center 16x16 window in the stereo disparity image at one time.

The system is able to verge on the target, in the middle of the images, extremely quickly with a rate of approximately 10 moves[1] per second, and quite accurately with *dense* and *reliable* stereo data input. It is observed that the system is fairly stable when the object is stationary, that is, the cameras can fixate on the un-moved target with only minor vergence movements due to round-off errors and noise. On the other hand, if the correlation process fails to return reliable and dense data, for instance, when the person standing in front of the cameras is wearing a white shirt with no texture, then it is no surprise that the robot head will frequently lose control and oscillate back and forth in a ridiculous and random fashion.

What we have learned from this experiment is that it is extremely easy to control the robot head with the type of data that does not have to be accumulated. In this case, we are not interested in how stereo changes over time, but only interested in how close an object is at a particular time instant. The performance of the tracking system certainly relies heavily on the accuracy of input data, especially in this experiment, where we do not interpolate the data of several stereo disparity images to obtain more accurate estimates of depth for tracking.

[1]When the reasoning system decides *not* to wait for the action system to finish the robot head's movement before continuing its processing.

5.1.1.3 Panning and Tilting without Verging

In this experiment, we allow our system to track an object moving in 3D by panning and tilting the robot head. In short, we will call this experiment the "pan-tilt" experiment. We are using the same input configuration for the datacube program as in the other experiments. From the previous motion detection experiment, we know that our system can correctly locate moving objects when there is no head movement. The goals of this experiment are to find out if we can correctly and reliably detect moving objects while the robot head is moving, and to examine the robustness of the overall system. The results collected will be analysed and further examined in the later discussions of the effectiveness and problems with our background optical flow cancellation method, the segmentation process, and the different selection algorithms.

The three optical flow accumulators, each responsible for dealing with one-third of the received data, can handle all the 64x64 optical flow images available at the rate of roughly 10 Hz. Our perception system will continuously add up the flow vectors regardless of whether or not the robot head is moving. The accumulated optical flow vectors are transferred to the segmentation processes of the tag programs on request. Each of the three segmentation processes will logically be responsible to work on one-third of the flow field. Such processes require an average of 0.85 second to complete the segmentation and statistical data collection duties. The connected components will then be passed on to the front program, which selects the visual target for the robot head to track.

The PRA loop will not repeat itself until the robot head has finished moving to a new location. The average time for executing one PRA loop is 1.8 seconds, which includes the time for the massive data being transferred among the Transputer processes, the time for messages being displayed by the different Transputer programs intended to report the status of execution, and the time for waiting the robot head to complete its movement. As a side note, it should be pointed out that printing too many messages from the Trollius operating environment back to the terminals on the Sun host will significantly increase the overall response time.

We have tested our motion tracking system on the simplest case scenario where there is only one object moving slowly in front of a *textured* background environment. We should point out that the background is *not artificial*, that is, the background is the usual cluttered laboratory environment. It is not especially arranged to have sufficient texture. The responses of the system have been fairly consistent, in that the robot head has been able to follow the moving object most of the time. The output of the segmentation process does indicate that our system can separate the moving object from the background even when the robot head is allowed to move, given that our background scene has purposely been designed to contain rich textural patterns so that the correlation output is *dense* with minimal mismatches. However, the system exhibits slightly different behaviour, as expected, in the passive following of the moving object, depending on the selection method used for picking out the visual target.

If the moving object occupies at least half of the image, then using the *largest area* selection method will enable the robot head to successfully track the moving object at most time, except that when the object slows down or attempts to change its direction, the correlation process will fail to detect consistent optical flow within the expected region, and the system will eventually lose track of the target due to the background optical flow that cannot be completely eliminated. However, we have observed that our system can easily recover from false alarms and "re-track" the moving object once the robot head has stopped moving.

If the *closest area* selection method is used to pick out the target object, then the stability of tracking the moving target will rely extremely heavily on the precison of optical flow computation, for the reason that without recognition, we are forced to use optical flow vectors to access stereo data for retrieving disparity values of the object. Round-off errors and noise are the major factors that we are generally getting poor performance with this technique, although it has been shown to be capable of following an object moving at a slow and constant speed. It has been observed that the background optical flow, however, has little effect on selecting the closest object as target.

Experimenting with multiple moving objects demonstrates that there are still a lot of problems remaining to be solved. One of the biggest questions is when the motion tracking system should shift attention to track another moving object. Depending on how the motion tracking system is defined to behave, there is certainly no single solution which will work perfectly in all situations.

The results of this pan-tilt experiment have revealed a number of problems with our current implementation, which will be discussed in details in the following sections. Our current motion tracking system is frankly far from being robust or stable. It is our hope to speed up the response time to at most 1 second per PRA loop. That is, we would like the robot head to make at least one move per second. This will definitely be a vast improvement to the current *near-real-time* performance.

5.1.2 Deficiencies, Problems, and What Cannot Be Done

5.1.2.1 Rigid Objects Assumption

The efficiency and reliability of a control system must take into consideration what it fails to accomplish, either expectedly or unexpectedly. We have described the assumptions which are made in connection to the limitations of our system. Specifically, we have assumed that we are dealing only with rigid moving objects, which will retain their shapes and structures throughout the period of being tracked. Non-rigid objects in general will not cause a great deal of difficulties for our system, except that changes in the rigidity of an object will be regarded as changes in locations by our reasoning system, which has no knowledge about any one specific object it is dealing with.

5.1.2.2 Disadvantages and Problems of Using Correlation Matching

The engine for detecting motion is correlation matching. As we have explained previously, a correlation range has to be selected for the matching operation. Due to the nature that this range is usually quite small as we desire to obtain fast output, we assume that objects are moving at a speed smooth and slow enough so that the matching operation can return reliable estimates for displacements. If an object is moving too fast according to the correlation range defined, then the correlation output will be inconsistent and extremely unreliable, and is undoubtedly unusable and impossible for the segmentation process to come up with a good description for driving the robot head to continue tracking. In other words, if correlation fails, then the tracking operation may immediately behave abnormally and unexpectedly.

The output of correlation is often corrupted by noise, mostly caused by quantization errors on edges, and sometimes the flickering of fluorescent lighting. The preciseness of the correlation output is inversely proportional to the resolution size of the images. Given the fact that images have been subsampled from the input size 512 to 64 for correlation, the fine details have been softened, and thus quantization errors increase with the coarser features in the images. The effect of noise visualizes as small motion patches on optical flow images that are frequently misinterpreted as moving objects. As a preventive measure against the distracting effect of noise, we decide to track objects only of a *reasonable* size by definition[2]. In other words, small connected components will be eliminated, and as a result, our motion tracking system is unable to track small moving objects, i.e., objects of sizes which are below the threshold defined for filtering out the small motion patches assumed to be produced by noise. This implementation error is inevitable in the attempt to stabilize motions of the robot head based on noisy input data.

In addition, the disparities of a moving object are retrieved from using optical flow vectors. Any error in correlation matching will seriously degrade the accuracy of accessing the stereo data, and thus handicap any other processes which rely on using stereo disparities for processing, such as the closest region selection method.

Correlation errors have also directly caused problems in the segmentation process. The accumulated correlation errors in the optical flow accumulation processes will not necessary cancel out, and this leads to an inconsistent and noisy flow field to be used in segmentation.

5.1.2.3 Problems with Background Optical Flow Cancellation

The background optical flow cancellation process is used with the assumption that the correlation process is capable of producing reliable matches at *every* point in the image. Unfortunately, this is not always the case when using real world data. As can be seen from our pan-tilt experiment, the background scene contains rich textural information for correlation. If it is the case that the robot head is moving but no optical flow is produced for the stationary background, for instance, the cameras are looking at a white wall, then our cancellation process will still attempt to subtract the expected background flow from everywhere of the flow field, not knowing that the correlation process has failed

[2]Possibly greater than 2% of the image size.

to return useful data. As a result, the motion tracking system will mistakenly identify the *compensated* motion flow field as moving objects, thus creating chaos and confusion for the selection routine.

5.1.2.4 Reliance on Connectedness

The effectiveness of our control theories relies heavily not only on the reliability of the correlation output, but also on the fact that the flow field generated by a moving object must appear to be contiguous. As we have no knowledge about any object, we are required to make the assumption that each connected component produced by the segmentation process corresponds to an individually moving object, which has no relation with any other moving object in view. The motion field corresponding to a rigid visual target may be split into several unconnected blobs if part of the target is being occluded. This leads to inaccuracies in tracking as the centroid of the target is computed from a partial motion field corresponding to that target. There is no simple solution to grouping unconnected motion blobs together without acquiring the service of an object recognition process similar to the one reported in [Lowe, 1987], which is mainly based on perceptual organization. Therefore, it is unavoidable for our motion tracking system to make such a mistake in computing the centroid.

On the other hand, two objects moving together with the same motion parameters may cause similar optical flow, and may be grouped into one single partition if their corresponding motion patches are connected in the optical flow images. It is possible to separate perceptually the two objects using stereo disparity if they are located at different depths. However, if this fails, and without any further information about the objects, then the single partition will be interpreted as one moving object by our motion tracking system.

5.1.2.5 Panning, Tilting, and Verging Simultaneously

Our current motion tracking system does not fully support the panning, tilting, and verging motions simultaneously at this time. The background optical flow cancellation process in our current system only deals with optical flow generated by the panning and tilting motions. Another cancellation process will be required to handle optical flow generated by the vergence motion separately, if we are to allow our robot head to move in all three axes at the same time to fixate a 3D location. It is rather easy to implement this additional cancellation process; however, it is our intention at this moment to study thoroughly the current setup, and to refine the control theories and implementation for better performance, before we proceed on to handling more complicated experiments.

The additional cancellation process for vergence can perhaps be implemented using the framework of the current cancellation process. It can also be based on the observation that optical flow generated from the right camera should in theory *cancel out* the optical flow generated from the left camera. The latter idea requires optical flow to be computed also from the point of view of the right camera.

Being able to verge while tracking allows us to use the zero-disparity feature to factor out the target object to be followed. We will certainly experiment with this idea once we have a more stable system.

5.1.2.6 Other Minor Issues

As for all motion tracking systems ever implemented, it should be obvious that if an object moves out of the range limit of the robot head, i.e., the object is out of reach, then it is impossible to continue the tracking operations on such object.

There are some other minor issues surrounding the performance of our motion tracking system, which deserve some attention, but are often ignored since they are in general not expected to have serious impact as compared to other issues that have already been or will be discussed. If we intend to increase the robustness of our system in every aspect, then much effort would be required on examining these issues in details.

For instance, we may need to investigate the effects on correlation matching when adjusting the focal length, aperture, shutter speed, gamma filters, and other controllable features of the stereo cameras. A CCD camera calibration process [Beyer, 1992] [Healey and Kondepudy, 1992] should be employed to estimate and eliminate noise in order to obtain more precise data. Backlash of the robot head often generates faulty input data. Applying more weight to the pan and tilt axes may help to eliminate the backlash effect, but it is still unknown if such action will slow down the head motion. However, we will soon be moving to a new robot head with significantly less backlash.

We may also need to adjust the parameters of the inverse kinematics equations in order to obtain smoother head movements. We should also find out ways to speed up the data transfer rate between two Transputer nodes within the Trollius environment or any other operating system we may use in future.

5.2 Additional Discussion

5.2.1 The Dumb Motion Trackers versus Our Motion Tracker

The absolute simplest tracking system is one that attempts to find some features, e.g., a white dot on a black background, on a single image to follow, without even computing for motion. Such system is undoubtedly useless when dealing with real world data. Another simple tracker can be made by sensing something moving by taking the difference of two images grabbed at different time, creating a binary image by thresholding the differences, and then finding the centroid for tracking. Although the system is fast, it is very difficult, if not impossible, to deal with noise or multiple objects. Computing displacements for motion using correlation matching, for example, will logically be the next option, if one wants to handle multiple moving objects.

The simplest motion tracking system is perhaps one that will observe the world only when the robot head is not moving, find out what is to be tracked, and then move the robot head without paying any attention to the changes in the world. The stop-and-look mechanism used in these systems excludes the idea of imaging while moving, and the use of sequential operations is extremely easy to implement and manage. Although this simple control system can be characterized as being a motion tracking system, it is one that is neither efficient nor robust. This system cannot continue to detect changes to any object while data is being analysed and the head is being moved to a new location. This would imply that the system can only act on a limited set of knowledge of the world, and the fact that grabbing an image should be a relatively fast operation as compared to data processing, the limited set of knowledge represents a very small proportion of changes that have occurred. It should be obvious that this system will experience great difficulty in following any target in a reliable and consistent manner with discrete and discontinuous input. For instance, there is no guarantee that the system can always grab the images that will identify the object generating the largest motion while the head is moving. Using assumptions or previous results definitely will not help much in filling for the missing information. We consider the idea of *blindfolding* oneself while in a busy states as a dumb behaviour, and in the extreme case unacceptable for use in a real-time system where responses are expected to reflect changes that are supposed to be smooth and steady.

The deficencies of not being attentive to the world at all time motivate our use of parallel processes, which have already been described in our control theories. Our idea is to continuously monitor the changes in the world so that we know exactly what had happened while our reasoning processes were busy working on something else.

5.2.2 Thresholding in Segmentation

Our goal of segmentation is to divide the optical flow field into separate connected components, each of which has a consistent set of optical flow values. Segmentation is known to be an interesting but hard problem, as reported by many researchers using different techniques [Jain, 1984] [Adiv, 1985] [Yamamoto, 1990] [Boult and Brown, 1991]. It is particularly difficult in our case when time constraint is a big factor preventing the use of sophisticated and complicated methods, which are usually time consuming.

The input to our segmentation process is accumulated *integer valued* optical flow. The threshold values unfortunately depend on the number of images used, as the error range of the round-off errors grows awkwardly and undesirably larger with more optical flow images being accumulated. Accumulated round-off errors is a major source of problems causing us to use relatively large threshold values in order to prevent the flow field to be overly divided into tiny motion patches. Correlation errors and noise also greatly affect the performance of our segmentation process.

Currently, we are using the magnitude of a vector as a criterion for determining connectedness, with the corresponding threshold set at 5 pixels. Smaller thresholds often misled the system to believe that the moving object is small in size, but in fact the object may be as large as one-third of the image size. Larger thresholds, on the other hand, generate the inverse effect of loosening the constraints for connectedness, but in general, cause less troubles for losing track of the target.

5.2.3 The Selection Methods

The problems we have with the background optical flow cancellation process have slowed down our investigation on the effectiveness of the different selection methods for picking out the visual target. As can be seen from our pan-tilt experiment, the largest region technique have already provided us a good starting point to work with, if we can somehow apply tricks to remove the background region by examining the bounding box of the moving region.

Some work is underway to find out how to make appropriate use of the "pop-out" effect created by the cancellation process. In theory, the magnitude of the background optical flow after cancellation should be significantly smaller than other moving objects. A selection method attempting to pick out the object that has the largest motion, a reasonable size, and a disparity value close to zero if vergence is also used in tracking, can perhaps exhibit stable behaviour if data is reliable to begin with.

5.2.4 Robustness versus Speed Tradeoff

The robustness and speed tradeoff problem exists in almost all real-time applications. If we want our motion tracking system to be robust, then huge amount of processing time is usually required for extensive computation. However, the robustness of a real-time system often depends on the response time and its ability to catch up with the pace of the real world. In order to speed up the response time, we usually need to cut down on the amount of computation, but that will undoubtedly decrease robustness, given that the computational resource is normally limited!

Knowing that it is difficult to have a system which is both extremely fast and robust, we need to value our goals and select the appropriate and proper behaviour to suit our objectives. We would not want a motion tracking system which spends an enormous amount of time "thinking" before responding. Such system would unlikely be classified as functioning in real-time. In other words, it is our belief that preciseness is in general less important than speed in a real-time system, considering the fact that the system can have many future chances for recovering and correcting any erroneous response made.

We should point out that a response made by the motion tracking system is the reaction to changes that have occurred while the system was busy working on the previous set of data. It would be wise to have a system which responds in a timely fashion, so that it would not fall behind seriously by analysing "out-dated" data. The faster the system can respond, the less the world has changed. Also, there will be less round-off errors and noise being accumulated with a faster responding system. Slow responding motion tracking systems can lose track of the target extremely easily, as objects may move out of the views of the cameras before they have been noticed.

5.2.5 Comparisons with Other Motion Tracking Systems

In this section, we will briefly comment on the differences between our current motion tracking system and some other different types of systems implemented elsewhere.

Following the largest moving object is a popular choice in motion tracking. A simple one camera motion tracking system has been implemented to pick the second largest moving area as target, assuming that the dominant motion to always be the motion of the background [Woodfill and Zabih, 1992]. Such system is running on a 16 kilo-processor Connection Machine; the tracking algorithm is sequential, and it is currently dealing with only one object. It is our opinion that there is no guarantee the background motion is always the largest in magnitude or in size. Our system takes the approach of attempting to cancel out the background motion, or at least minimizing its effect causing it to be the least significant motion, i.e., all moving objects should have flow values larger than background motion after cancellation. Eventually, we will pick the target based primarily on the magnitude of motion, instead of the size of the region.

Coombs' approach [Coombs, 1992], as already discussed in Chapter 2, is to first verge the cameras and then use a Zero-Disparity Filter to pick out the target. Their system is quite robust and has good performance with a prediction module. It is not clear, however, how such system can deal with multiple objects, or to shift attention. The approach taken at the University of Rochester is to throw away, or filter out, irrelevant data or useless information as quickly as possible. Our datacube program produces stereo data, so it would be simple to implement ZDF in our system; we have decided to attack a more difficult problem, using motion data. Our current system uses more output data, and therefore, has a much better record of the motion paths of *all* moving objects in view for further analyses.

An attentive control system [Clark and Ferrier, 1988] has been implemented to track features. As described in their report, shifts in focus of attention is accomplished by using a saliency map and by altering of feedback gains applied to the visual feedback paths in the position and velocity control loops of the binocular camera system. Since we do not have any knowledge about any object we are dealing with, our system simply uses motion field to drive attentional processing.

The KTH Head has 13 degrees of freedom and 15 different motors, simulating the essential degrees of freedom in mammalians [Pahlavan *et al.*, 1992]. Their current work suggests the integration of low-level ocular processes for fixation, and the use of cooperative vergence-focussing to assist the matching process. It is difficult for us to experiment with their ideas and the focussing algorithm, as our current LCI head does not have motorized focus control or zoom lenses.

Chapter 6

Conclusions and Future Directions

6.1 Concluding Remarks

The primary goal of motion tracking is to keep an object of interest, generally known as the visual target, in the view of the observer at all time. One of our objectives in this project is to implement a simple gaze control system on our robotic camera head system to demonstrate that we can find out where an object is without knowing what the object is. Our decision of not having, in our system, any particular knowledge about any object prevents the use of a feature-based object recognition process for tracking. As an alternative, tracking can be driven by changes perceived from the real world, and in this project, we have chosen to use displacements as the major source of directing signals.

In this thesis, we have described a set of control theories to track a moving object in real-time with a three degrees of freedom robot head. The recent advances in computer hardware, exemplified by our Datacube MaxVideo 200 system and a network of Transputers, make it possible for researchers to perform image processing operations at video rates, and to implement real-time systems with input images obtained from video cameras. We have taken advantages of our powerful equipment at LCI, and have developed a passive motion tracking system which reacts to changes in the surrounding environment by panning, tilting, and verging the robot head.

The control scheme of our motion tracking system is based on the Perception-Reasoning-Action regime. Our idea is to use parallel processes, which communicate with one another via message passing, to actively monitor changes in the environment, to process displacements and select the visual target, and to control the movements of the robot head. The amount of data to be processed and the amount of computation have been designed to be as minimal as possible, so that the system can react to the changes perceived in a timely manner, and can keep up with the pace of the world.

We have described an elegant approach of using an active monitoring process together with a process for accumulating temporal data to allow different hardware components running at different rates to communicate and cooperate in a real-time system working on real world data. We have no control on changes in the real world, and therefore, there should be no delay to the processes producing data to reflect such changes, even when some other processes are in busy states. The stream of output data can be stored and grouped together for the busy processes to retrieve at a later time.

We have also described a cancellation method to reduce the unstable effects of background optical flow generated from ego-motion, and create a "pop-out" effect in the motion field in the sense that after cancellation, the motions of the moving objects should be significantly larger than the background motion. A simple segmentation process has been developed to partition the motion field into separate connected components based on the consistency of optical flow vectors; each component is interpreted as a moving object. A few selection routines have been implemented to pick out a target. No single selection method is sufficient to be applied in all situations. The one to be used really depends on how the motion tracking system is defined to behave.

The problem of tracking moving objects using only displacements has shown to be difficult and challenging. The results of various experiments provide us with an insight of the difficulties of tracking without any knowledge of the world and the objects. Round-off errors, quantization errors, and correlation errors have seriously affected the performance and degraded the robustness of our motion tracking system. Nevertheless, we are able to track a moving person by panning and tilting the robot head in approximately 1.8 seconds, and fixate an object by verging the stereo cameras at a rate of roughly 10 moves per second.

The system we have described in this thesis can assist other vision tasks, such as object recognition, by always keeping the object of interest in view for studying. The motion path of the object being tracked can easily be recorded, and may allow a *motion path based* object recognition system to be developed.

The central assertion of this thesis is that we can track an object prior to such object is being identified. Displacement has proven to be an important cue for tracking. It is arguable as to whether or not optical flow and stereo disparity can be classified as *primary* visual cues. Nevertheless, they provide the sufficient information for our motion tracking system to follow an object moving in the three-dimensional space that is accessible to the cameras.

6.2 Future Work and Possible Improvements

There are a number of items and ideas on our list for improving the performance and for expanding the functionalities of our current motion tracking system. The following list also covers areas which we have to address in future research.

- We should include prediction in future versions of our motion tracking system. A prediction process, as we have already described, allows us to move the robot head even when some reasoning process is busy analysing data. A Kalman filter can be used to implement the prediction process. The control theories will have to be revised to accommodate the idea of parallelizing the operations in the reasoning and action systems. The prediction process can be very simple, as it is only concerned with the centroid of the target.

- We need to improve the correlation technique used in the datacube program so that the displacement output is more reliable. Neighboring displacement values should be examined when there is a need to resolve ambiguities. We should consider returning the confidence measures within the correlation process to the Transputer processes, so that displacements can be better managed on the receiving end.

- Knowing that the correlation process will not guarantee reliable data at all time. We should find out ways to make use of the edges and Laplacian of Gaussian images returned by our datacube program, so that more information can be available to the segmentation process. Techniques such as those used in [Poggio *et al.*, 1988] [Gamble and Poggio, 1987] can perhaps be applied to our system.

- We need *sub-pixel* accuracy for displacements in order to help eliminating the problem of round-off errors and quantization errors being grossly built up in the accumulation processes. This will help the background optical flow cancellation process to work to its potential. However, it is unclear how sub-pixel accuracy can be obtained or represented in our current implementation.

- One idea to cut down the amount of computation and to speed up the response time significantly is to use previously computed data to assist the picking of the visual target, as we already have the expectation of where the target will end up. The computation of stereo disparity can be performed on a demand or *feedback* basis, so that the current problem of accessing stereo data using optical flow vectors as guides can be eliminated.

- Experiments have shown that correlation matching performed on 128x128 images can return better and more consistent optical flow. We should find out ways to make the datacube program return 64x64 output images, but allow the correlation process to work on images with larger resolutions for better matches.

- Some work has already been started on optimizing the Transputer programs. Specifically, we want to discover routes to cut down the response time, and to speed up the two-way data transfer rate among the Transputer processes being executed on different Transputer nodes. We may need to upgrade the operating system running on the Transputer nodes, or use faster hardware, for example, Texas Instrument TMS320C40 processors.

- The current LCI robot head will be upgraded to use gears and motors that have significantly less backlash.

- If we can create the special stereo mapping table for background optical flow cancellation using a separate camera calibration process, then we may use the vergence angle for indexing the special table if all three axes can move simultaneously, and thus allows a better approximation of the expected background flow to be used.

- We need a more robust segmentation algorithm, which probably require more constraints for partitioning the optical flow field. The prediction process may be able to provide useful data for filtering out areas which are unlikely to contain the target being followed, so that the overall response time can be reduced.

- There are a few advantages for computing optical flow from the view points of *both* cameras, as opposed to the current setup of using just the left camera.

  - The two sets of optical flow data provide more information to allow a consistency displacement check.

  - We can implement the additional cancellation process for vergence, so that we can pan, tilt, and verge the robot head at the same time. In theory, optical flow due to vergence from the left and from the right cameras should cancel out.

  - More importantly, we can process true three-dimensional data. Optical flow from the left camera and the right camera, together with stereo disparity, enable us to establish a field of 3D motion vectors. This allows the segmentation process to work with a more consistent and informative flow field. In addition, we can compute the 3D motion parameters of the moving objects for better analyses. Unfortunately, it is very unlikely that these features will be implemented in the real-time system at this moment, since the extra computation will be extremely time consuming. In any case, they are certainly worth looking into in the future.

Bibliography

[Adiv, 1985] G. Adiv. Determining Three-Dimensional Motion and Structure from Optical Flow Generated by Several Moving Objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7(4):384–401, 1985.

[Anandan, 1989] P. Anandan. A Computational Framework and an Algorithm for the Measurement of Visual Motion. *International Journal of Computer Vision*, 2:283–310, 1989.

[Balasubramanyam and Snyder, 1988] P. Balasubramanyam and M. A. Snyder. Computation of Motion in Depth Parameters: A First Step in Stereoscopic Motion Interpretation. In *Proc. 1988 DARPA Image Understanding Workshop*, pages 907–920, 1988.

[Ballard and Brown, 1992] D. H. Ballard and C. M. Brown. Principles of Animate Vision. *Computer Vision Graphics and Image Processing*, 56(1):3–21, July 1992.

[Barron *et al.*, 1992] J. L. Barron, D. J. Fleet, and S. S. Beauchemin. Performance of Optical Flow Techniques. TR–299, The University of Western Ontario, July 1992.

[Beyer, 1992] H. A. Beyer. Accurate Calibration of CCD-Cameras. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition, 1992*, pages 96–101, 1992.

[Boult and Brown, 1991] T. E. Boult and L. G. Brown. Factorization-based Segmentation of Motions. In *Proc. IEEE Workshop on Visual Motion, 1991*, pages 179–186. IEEE, 1991.

[Brooks, 1987] R. A. Brooks. Intelligence without representation. In *Proceedings, Workshop on the Foundations of Artificial Intelligence*, 1987.

[Brown, 1989] C. M. Brown. Prediction in Gaze and Saccade Control. TR–295, University of Rochester, May 1989.

[Bulthoff *et al.*, 1989] H. Bulthoff, J. J. Little, and T. Poggio. A parallel algorithm for real-time computation of optical flow. *Nature*, 337:549–553, February 1989.

[Christensen, 1992] H. I. Christensen. The AUC robot camera head. In *SPIE Vol. 1708 Applications of Artificial Intelligence X: Machine Vision and Robotics*, pages 26–33, April 1992.

[Clark and Ferrier, 1988] J. J. Clark and N. J. Ferrier. Modal Control of an Attentive Vision System. In *Proc. 2nd International Conference on Computer Vision*, pages 514–523, 1988.

[Coombs and Brown, 1992] D. Coombs and C. Brown. Real-time Smooth Pursuit Tracking for a Moving Binocular Robot. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition, 1992*, pages 23–28. IEEE, 1992.

[Coombs, 1992] D. J. Coombs. *Real-time Gaze Holding in Binocular Robot Vision.* PhD thesis, University of Rochester, Rochester, New York, June 1992.

[Crowley *et al.*, 1992] J. L. Crowley, P. Bobet, and M. Mesrabi. Layered Control of a Binocular Camera Head. In *SPIE Vol. 1708 Applications of Artificial Intelligence X: Machine Vision and Robotics*, pages 47–61, April 1992.

[Drumheller and Poggio, 1986] Michael Drumheller and Tomaso Poggio. On Parallel Stereo. In *Proc. IEEE Conf. on Robotics and Automation, 1986*, pages 1439–1448, Washington, DC, 1986. Proceedings of the IEEE.

[Ferrier, 1992] N. J. Ferrier. The Harvard Binocular Head. In *SPIE Vol. 1708 Applications of Artificial Intelligence X: Machine Vision and Robotics*, pages 2–13, April 1992.

[Fua, 1991] P. Fua. A Parallel Stereo Algorithm that produces Dense Depth Maps and preserves Image Features. Technical report 1369, INRIA, 1991.

[Gamble and Poggio, 1987] E. B. Gamble and T. Poggio. Visual integration and detection of discontinuities: The key role of intensity edges. A.I. Memo No. 970, MIT AI Laboratory, October 1987.

[Gibson, 1979] J. J. Gibson. *The Ecological Approach to Visual Perception.* Houghton Mifflin Co., Boston, MA, 1979.

[Healey and Kondepudy, 1992] G. Healey and R. Kondepudy. CCD Camera Calibration and Noise Estimation. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition, 1992*, pages 90–95, 1992.

[Heeger *et al.*, 1991] D. J. Heeger, A. D. Jepson, and E. P. Simoncelli. Recovering Observer Translation with Center-Surround Operators. In *Proc. IEEE Workshop on Visual Motion, 1991*, pages 95–100. IEEE, 1991.

[Jain, 1984] R. C. Jain. Segmentation of Frame Sequences Obtained by a Moving Observer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(5):624–629, September 1984.

[Jenkin *et al.*, 1992] M. Jenkin, E. Milios, and J. Tsotsos. TRISH: The Toronto-IRIS Stereo Head. In *SPIE Vol. 1708 Applications of Artificial Intelligence X: Machine Vision and Robotics*, pages 36–46, April 1992.

[Krishnan and Stark, 1977] V. V. Krishnan and L. Stark. A Heuristic Model for the Human Vergence Eye Movement System. *IEEE Transactions on Biomedical Engineering*, 24(1):44–49, January 1977.

[Little and Gillett, 1990] J. J. Little and W. E. Gillett. Direct Evidence of Occlusion in Stereo and Motion. TR-90–5, UBC Dept. of Computer Science, Vancouver, BC, 1990.

[Little *et al.*, 1991] J. J. Little, R. Barman, S. Kingdon, and J. Lu. Computational Architectures for Responsive Vision: the Vision Engine. TR-91–25, UBC Dept. of Computer Science, Vancouver, BC, 1991.

[Lowe, 1987] D. G. Lowe. Three-Dimensional Object Recognition From Single Two-Dimensional Images. *Artificial Intelligence*, 31:355–395, 1987.

[Marr and Poggio, 1976] David Marr and Tomaso Poggio. Cooperative Computation of Stereo Disparity. *Science*, 194(4262):283–287, October 1976.

[Matthies *et al.*, 1988] L. Matthies, R. Szeliski, and T. Kanade. Kalman filter-based algorithms for estimating depth from image sequences. In *Proc. 1988 DARPA Image Understanding Workshop*, 1988.

[Nelson, 1990] R. C. Nelson. Qualitative Detection of Motion by a Moving Observer. TR–341, University of Rochester, April 1990.

[Nelson, 1991] R. C. Nelson. Introduction: Vision as Intelligent Behavior – An Introduction to Machine Vision at the University of Rochester. *International Journal of Computer Vision*, 7(1):5–9, 1991.

[Olson and Coombs, 1991] T. J. Olson and D. J. Coombs. Real-time Vergence Control for Binocular Robots. *International Journal of Computer Vision*, 7(1):67–89, 1991.

[Pahlavan and Eklundh, 1992] K. Pahlavan and J-O. Eklundh. Heads, Eyes, and Head-Eye Systems. In *SPIE Vol. 1708 Applications of Artificial Intelligence X: Machine Vision and Robotics*, pages 14–25, April 1992.

[Pahlavan *et al.*, 1992] K. Pahlavan, T. Uhlin, and J. Eklundh. Integrating Primary Ocular Processes. In *ECCV*, pages 526–541, 1992.

[Peleg and Rom, 1990] S. Peleg and H. Rom. Motion Based Segmentation. In *Proc. 10th International Conference on Pattern Recognition*, pages 109–113. IEEE, 1990.

[Poggio *et al.*, 1988] T. Poggio, E. B. Gamble Jr., and J. J. Little. Parallel integration of vision modules. *Science*, 242(4877):436–440, October 1988.

[Pretlove and Parker, 1992] J. R. G. Pretlove and G. A. Parker. A light weight camera head for robotic-based binocular stereo vision: An integrated engineering approach. In *SPIE Vol. 1708 Applications of Artificial Intelligence X: Machine Vision and Robotics*, pages 62–75, April 1992.

[Robinson, 1968] D. A. Robinson. The Oculomotor Control System : A Review. In *Proceedings of the IEEE, volume 56*, pages 1032–1049, 1968.

[Shio and Sklansky, 1991] A. Shio and J. Sklansky. Segmentation of People in Motion. In *Proc. IEEE Workshop on Visual Motion, 1991*, pages 325–332. IEEE, 1991.

[Singh, 1991] A. Singh. Incremental Estimation of Image-Flow Using a Kalman Filter. In *Proc. IEEE Workshop on Visual Motion, 1991*, pages 36–43. IEEE, 1991.

[The Ohio State University, 1991] Research Computing The Ohio State University. Trollius 2.1 User's Reference and C Reference Manual, March 1991.

[Thompson and Pong, 1987] W. B. Thompson and T. C. Pong. Detecting Moving Objects. In *Proc. 1st International Conference on Computer Vision*, pages 201–208. IEEE, 1987.

[Waxman and Duncan, 1986] A. M. Waxman and J. H. Duncan. Binocular Image Flows: Steps toward Stereo-Motion Fusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):715–729, November 1986.

[Woodfill and Zabih, 1992] J. Woodfill and R. Zabih. Using Motion Vision for a Simple Robotic Task. In *AAAI Fall Symposium Series: Sensory Aspects of Robotic Intelligence*, pages 152–159, 1992.

[Yamamoto, 1990] M. Yamamoto. A Segmentation Method Based on Motion from Image Segmentation and Depth. In *Proc. 10th International Conference on Pattern Recognition*, pages 230–232. IEEE, 1990.

[Zhang and Faugeras, 1992] Z. Zhang and O. D. Faugeras. Three-Dimensional Motion Computation and Object Segmentation in a Long Sequence of Stereo Frames. *International Journal of Computer Vision*, 7(3):211–241, 1992.