

# Towards an Active Visual Observer

Tomas Uhlin, Peter Nordlund, Atsuto Maki and Jan-Olof Eklundh

Computational Vision and Active Perception Laboratory (CVAP)

Department of Numerical Analysis and Computing Science

KTH (Royal Institute of Technology), S-100 44 Stockholm, Sweden

Email: tomas@bion.kth.se, petern@bion.kth.se, maki@bion.kth.se, joe@bion.kth.se

March 1995

## Abstract

We present a binocular active vision system that can attend to and fixate a moving target. Our system has an open and expandable design and it forms the first steps of a long term effort towards developing an active observer using vision to interact with the environment, in particular capable of figure-ground segmentation. We also present partial real-time implementations of this system and show their performance in real-world situations together with motor control. In pursuit we particularly focus on occlusions of other, both stationary and moving, targets, and integrate three cues to obtain an overall robust behavior, ego-motion, target motion and target disparity.

We argue that an active vision system must be open, expandable, and operate with whatever data are available momentarily. The integration of many cues and concurrent modules, rather than sophisticated recovery mechanisms, forms a pervading theme. We stress that an active system must be equipped with means and clues to change its attention, while it otherwise is purely reactive. This system is therefore equipped with motion detection for changing attention and pursuit for maintaining attention, both of which run concurrently.

Keywords: Active and Real-Time Vision, Integration of Modules and Cues

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Related Work . . . . .	1
<b>2</b>	<b>Vision Systems and Active Vision</b>	<b>2</b>
2.1	Desired Properties . . . . .	3
2.2	Objectives . . . . .	3
<b>3</b>	<b>The Visual Front-End</b>	<b>4</b>
<b>4</b>	<b>System Description</b>	<b>4</b>
4.1	Attention and Smooth Pursuit . . . . .	5
4.1.1	Motion Detection . . . . .	6
4.1.2	Target Segmentation . . . . .	8
4.1.3	Disparity Segmentation . . . . .	9
4.2	Integration . . . . .	9
<b>5</b>	<b>Real-Time Implementation</b>	<b>10</b>
5.1	The Real-Time Pursuit Mechanism . . . . .	10
5.2	The Real-Time Motion Detection Mechanism . . . . .	12
<b>6</b>	<b>Experiments and Results</b>	<b>12</b>
6.1	Experimental Setup . . . . .	13
6.2	Real-Time Pursuit with Motor Control . . . . .	14
6.3	Handling Occlusions . . . . .	14
6.4	Occlusion and Other Moving Objects . . . . .	19
6.5	Expanding Targets . . . . .	19
6.6	Rotating Targets . . . . .	20
6.7	Real-Time Motion Detection with Motor Control . . . . .	21
<b>7</b>	<b>Discussion</b>	<b>24</b>
<b>A</b>	<b>Theory</b>	<b>25</b>
A.1	Brightness Constancy and Affine Image Velocity . . . . .	25
A.1.1	Brightness Constancy . . . . .	25
A.1.2	Residual Normal Velocity Calculation . . . . .	25
A.1.3	Solving for the Affine Image Velocity . . . . .	25
A.2	Phase-Based Disparity Estimation . . . . .	26

# 1 Introduction

Vision is a sense by which seeing creatures acquire information about a dynamically changing environment and thereby guide many of their behaviors and actions. A key mechanism in this process is fixation. Several researchers have argued convincingly for its importance in active machine vision, see e.g. Ballard (1991), Murray *et al.* (1993) and Pahlavan *et al.* (1993). Despite reported successful attempts proposed computational approaches have been limited in scope. Experiments on realistic and complex scenes running in real-time are rare. Even though references to biological vision often are made, we are clearly far from the abilities that can be observed in nature, both with respect to performance and robustness. Of course, there are technological reasons for this: cameras are rather poor sensors compared to eyes and available computers are not capable to perform the necessary real-time computations. However, we contend that another reason for the limited success, both on the specific problem of dynamic fixation and, more generally, in developing machine vision systems that even in a weak sense "see", is that the problem generally is being addressed from a rather narrow perspective. Techniques for fixation and tracking are often based on a single cue and various restrictions on the motion, and sometimes also on calibration. The focus has often been on single algorithms in isolation and the theories underlying them. Hence there is a host of results on computing optic flow and disparities, and algorithms for vergence and stabilization, but few reports on how useful these algorithms are in a broader context.

We argue that such methods, however elaborate they may be, are insufficient to provide robust behaviors in a complex and dynamically changing environment. Instead we propose a systems oriented approach in which multiple cues are used concurrently. In this way we can utilize the fact that the world is rich on information and rely on the cues that are actually available at a given instant to get a *well-behaved system*. Our overall approach aims at real-time implementations, since we also consider continuous operation over time as another prerequisite for a seeing system. At the moment we are therefore limited in our choice of algorithms. However, our methodology is general and based on an open architecture. Therefore, more sophisticated algorithms as well as additional cues can be incorporated in the current framework as they become feasible to implement. Notably, such extensions are facilitated by our use of a common visual front-end layer for the retinotopic processing providing input to the ensuing computations of cues. Our long-term goal is to develop an active visual observer. In this paper we consider the most basic behavior of such a system, the ability to fixate, built up of a pursuit and an attention mechanism.

## 1.1 Related Work

Notwithstanding our remarks from above many researcher have presented work towards the same goal of achieving robust pursuit. Coombs and Brown (1991) were among one of the first to show real-time (15Hz) performance in an active vision system. This work has some shortcomings in what targets can be pursued, the most important being that it relies on zero-disparity filtering to extract the target. This means that the system is unable to pursue things that are far away or are e.g. crawling on the floor. It also seems to have difficulties to deal with scenes containing objects at a variety of disparities, see Taylor *et al.* (1994). However, it represents one of the few reported binocular systems. Moreover the way it is implemented it ensures a correct integration of vergence and pursuit in that the target which is verged upon will always be the same as what is being pursued.

Murray *et al.* (1993) have demonstrated an active vision system which performs monoc-

ular pursuit in real-time (25Hz). The presented implementation relies on the extraction and tracking of corners and a question is how this performs in very patterned environments. The features could of course be any features, but then some method for automatic selection of features must be included for generality. A special feature of their system is the detection of independently moving targets, in which also an estimated target speed is included, such that a saccade is performed to a correct target location with the correct initial tracking speed. The motion detection however assumes that the background motion comes from camera rotations only and that camera rotations are known.

Tölg (1992) implemented a biologically inspired monocular pursuit system which performs pursuit in real time (2-3Hz). Tölg's system is based on flow computation and also builds a dynamic shape model of the pursued target and is in this way similar to the work presented here. However, some assumptions are made that are undesirable. Mainly fronto-parallel movement is assumed, and pursuit is performed only in the horizontal direction. Also some details in the algorithms are not suitable for fast hardware implementations. Tölg does not present the performance in the presence of other moving targets. Promising in Tölg's work is the system design, where a modular and hierarchical framework lets the system form a basis for building a more advanced system. Novel features are that the system computes ego-motion from the image, and catch-up mechanisms are incorporated to give generality and robust behavior.

More recently Prokopowicz *et al.* (1994) described an approach to visual tracking in which one out of several tracking modules is chosen depending on the situation at hand. The spirit of this is somewhat related to ours, but the emphasis is on a higher level — also cognitive aspects are included — and the low-level problems of the active observer are not considered.

Sundareswaran *et al.* (1994) and related papers discuss visual servoing based on derived motion parameters, which also forms part of our work. However no integrated system is presented and only simple motion patterns can be treated one at a time. These comments apply also to Andersen and Christensen (1994), which indeed consider multiple cues in a variation of our approach in Pahlavan *et al.* (1993).

There is of course also a large amount of work on individual techniques used here, for instance the extensive work on optic flow segmentation as well as on disparity computations. We do not claim to make any contributions on these problems per se. What we want to show is rather that even simple and sometimes coarse algorithms working together can provide a robustly functioning system. Contributions that in particular have inspired this work are among others Irani and Peleg (1993), Wang and Adelsen (1994).

Finally, since we are applying a systems oriented approach to real-time problems, the highly successful work by Dickmanns and his colleagues and Thorpe and his colleagues on real-time vehicle guidance must be mentioned, see e.g. Dickmanns *et al.* (1993) and Thorpe *et al.* (1994). Although on a specific problem, these efforts have much in common with our work.

## 2 Vision Systems and Active Vision

As stated in the introduction much of the vision research is directed towards developing theories and algorithms for specific problems and situations. Although this might be well motivated from a scientific point of view, the implementations generated are often designed to function optimally in isolation and little effort is spent on how they should function together with other algorithms. Implementations often take one image, or a sequence of images, a stereo

pair or what have you, and produce an output.

Scientifically we obtain limits on what can be computed with the given information in the situation at hand. However, the importance of these limits does not become clear until the computed output is used for something. Notably, we are in computer vision trying to develop “seeing systems”, not observing existing systems.

There are also practical implications of such approaches. Existing implementations usually contain no hooks for inclusion of other cues or knowledge sources. A particular consequence of this is that systematic experimentation becomes difficult, because extensions usually require reimplementations. For instance, a particular visual module should at least allow input from a general preprocessor, a visual front-end, if overall system performance is essential and extendibility is desired. Recent work on motion, stereo and texture supports such an idea, see e.g. Gårding and Lindeberg (1994).

## 2.1 Desired Properties

Whatever architecture one uses, a system of the type we discuss should be modularly built to be extendible. It cannot be designed to include everything from the beginning, or be limited to do just what is implemented at the moment. For efficiency reasons the system should be designed to share data, see Section 3. The overall behavior of the system must not depend heavily on any single part of the system unless this part is correspondingly more robust. A system’s performance is no better than the performance of its weakest part. It is important to remember that the system will never be complete, there will always be situations when the performance is not good enough for a given task. This must be taken into account in the design so that, as knowledge increases, more competent and specialized<sup>1</sup> components can be incorporated.

## 2.2 Objectives

In our attempt to develop an active visual observer we have so far implemented what could be called its basic skills. These are functionalities that are natural and desirable for any active vision system, skills that are directly related to attention and fixation. The design here is strictly guided by the ideas outlined earlier in that the system is meant to function as the bottom layer of a larger system, where new functionalities can be included which can make use of what is there and the already existing system can benefit from what is included.

The way the system will function will of course in the end depend on its goals. At this stage we are considering behaviors based solely on bottom-up information, although the algorithms are tightly connected through feed forward and feedback mechanism, embodying the cue integration.

In summary, the system currently only has the innate goal of attending and visually stabilizing moving targets, while it is itself moving both its body (platform) and its head. We regard this as the basic behavior necessary for most other higher level processes the active observer may wish to do. In particular, our work so far clearly demonstrates how the integration of monocular, binocular and motion cues resolves much of the figure-ground problem: the targets found are objects in the scene. This indicates that an integrated use of multiple low-level cues in general can be applied to come to grips with the figure-ground

---

<sup>1</sup>Specialized, because if it is overall more competent, the previous component can simply be replaced.

problem, without the often used high-level information<sup>2</sup>.

The target hardware for our work is the Head-Eye system described in Pahlavan (1993), now mounted on a mobile platform. The current implementation comprises a general system for pursuit, described in detail in Section 5.1. The major parts of the current system are,

- selecting a target
- control of the system for saccade and pursuit
- measuring the speed of the target for pursuit
- measuring and selecting a disparity for pursuit

As reported e.g. in Pahlavan *et al.* (1993) we have already realized certain high performance experiments on our Head-Eye system. Our continued work aims at further developing this system by adding new features in an integrated way, keeping the basic components with adequate performance, and applying the overall systems perspective.

### 3 The Visual Front-End

Current research on early processes in computer vision to increasing extent seems to converge on certain basic principles. Generally speaking most approaches rely on the computation of directional derivatives of low orders and combinations or functions thereof, computations performed at several different scales. The manner in which this is done varies: some approaches use Gaussian or Gabor filters, other tunable filters of a more general nature, wavelets or more sophisticated anisotropic models. It is beyond the scope of this paper to discuss these different techniques in detail, we refer the reader to the literature on the topic. However, we note that such methods in general are indeed very appropriate for developing systems of the type we are aiming at. By implementing a first layer of retinotopic processing, a Visual Front-End in the terminology as proposed by Koenderink and van Doorn(1987), we obtain a highly efficient implementation at the same time as we can base our low level computations on state-of-the-art techniques. We also get a scalable design, which allows the extensions we foresee, without cumbersome additions.

More precisely, by computing a set of low order derivatives at multiple scales in a VFE layer, we obtain output that can be shared by all our subsequent modules to derive monocular, binocular and motion cues at later stages. Without arguing for or against whether such a model provides the most reasonable architecture for a computer vision system, we observe that at least it maps well onto current hardware. Existing pipeline and signal processors are well suited for such retinotopic computations, but less so for the more general and less image oriented ensuing computations, which usually are performed in coarse grained parallelism. In the end we may want to develop other pathways for more direct computations of, say time-to-collision or certain scene characteristics, but currently we favor the VFE structure. The details of our approach will be described in the later sections.

### 4 System Description

Our aim is, as mentioned above, the design and implementation of a fully autonomous mobile observer. So far fundamental skills in terms of fixation, target pursuit and target discrimination, have been implemented. The basic system is at the moment implemented partly in

---

<sup>2</sup>See (Grimson et al., 1994) for a similar argument, even though such information when available of course is very powerful.

real-time on an existing mobile platform, and partly as a post-processing stage working on images taken in real-time using the former processes.

The full system includes the integration of three cues for target selection and target discrimination. These are used by the moving observer to smoothly pursue moving or stationary targets binocularly while maintaining vergence. Mechanisms for discovering moving targets also form integral and vital parts of the system, since that provides means of attention, without which clues to changing the state of the system do not exist. In other words, any active vision system must have two components, one to maintain attention, and another one to find and select new locations to attend to. Moreover, these must function in parallel, while otherwise the system would be purely reactive and without choices. We have implemented these two components in the form of a pursuit and a motion detection mechanism<sup>3</sup>, thus obtaining what we believe is the most basic behavioral level for an active observer. This is shown schematically in Figure 1.

**Terminology and Comments** Throughout this section we will use the term *target* and with this we mean the portion in the image which is currently attended and supposed to represent a scene-object. A *target model* is used, which is coded as an image mask, including pixels that are considered as belonging to the target. All segmentation modules produce similar masks after a thresholding step which are used for integration. References will often be made to Appendix A and the reader who is not familiar with image velocity and disparity calculations could benefit from reading this first.

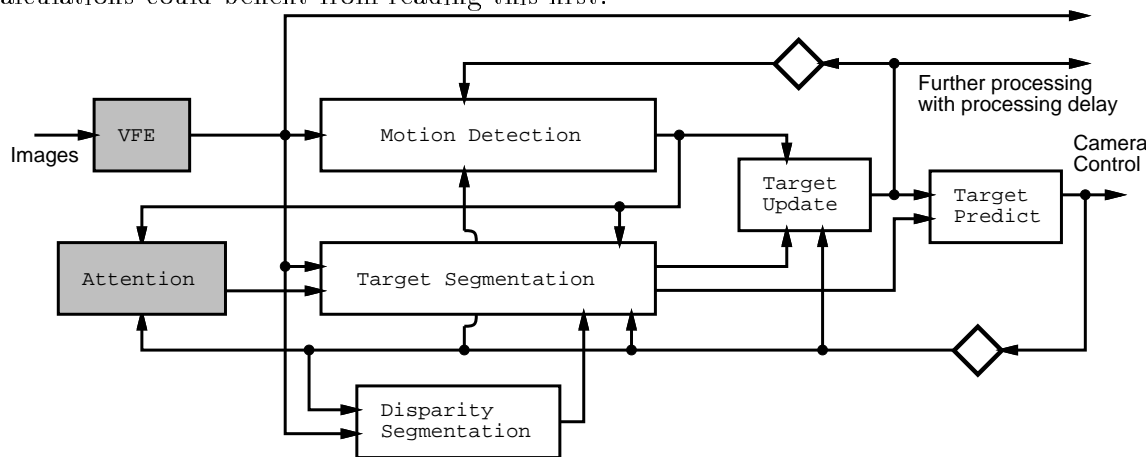


Figure 1: The system implementation is shown schematically. (The diamonds indicate a one frame delay in the feedback.)

#### 4.1 Attention and Smooth Pursuit

A key feature of the system lies in its ability to smoothly pursue an arbitrary target that is described by its location, extent and visual appearance. We would like to stress here that no apriori knowledge about the targets visual appearance, i.e. texture or shape, is built into the system, although we indeed leave open that additional constraints or knowledge can be included dynamically as information becomes available, and similarly that such information can be excluded as it becomes obsolete. The features that are seen, are assumed by the pursuit module to change smoothly. Non smooth changes will trigger attentional mechanisms.

<sup>3</sup>Currently, motion provides the only cue to changing attention, but other cues can easily be incorporated.

As the basis for this implementation we have chosen a coarse to fine correlation scheme similar to the one reported in Pahlavan *et al.* (1993), see Section 5.1. This technique works very well when no occlusions are present. We will here present an extension handling occlusions as well.

In order to take care of occluding objects and distracting things in the background, motion detection is integrated to filter out parts of the scene that can be parts of the target, namely those that are moving. Such a mechanism is provided in the system by a connection from **Motion Detection** to **Target Segmentation**, see Figure 1. We will see later in the experiments that this is not enough in many cases when there are occluding objects that are themselves moving, since they will also be detected as moving. Disparity, which is an essential component in our binocular system can in these situations aid in depth discrimination by also providing a clue to where occluding objects may lie.

**Issues on Iterative Algorithms and “Anytime Vision”** In the motion literature quite a few iterative algorithms have been presented. In a real-time active system it is of importance that the modules that control the system respond within a given time, although this time demand will vary from task to task, and no general statement can be made about a maximum time without knowledge of the task and the type of control involved. What can be said, though, is that this time constraint applies to all steps in the processing, unless the system specifically deals with the modules that involve slower computations. More specifically, a module that runs in real-time and in turn depends on the input from a slower module must explicitly deal with the fact that it is going to receive *old data*. For instance, many algorithms in the motion segmentation literature involve an initialization step to produce an initial segmentation in the beginning of a sequence, a step which is many times slower than the ensuing updating of the segmentation as the sequence evolves. Unless such algorithms explicitly deal with the initial delay of “bootstrap”, the effective real-time of such algorithms in an active vision system is that of the bootstrap time and not the consequent higher processing rate.

We believe in the notion that information should be made available as soon as possible, so that the system can react to the stimuli in time. Even though an early reaction may be wrong, it is often better than no reaction at all. Hence, the “bootstrap” should be made over time while the system is functioning, and data be made available during this phase so that the system always has access to current data, even if in the beginning it is not of the best quality. In general any part of an active real-time system should have access to data whenever it is needed and modules involved in the system should be designed to provide this. We refer to this as *Anytime Vision*<sup>4</sup>.

#### 4.1.1 Motion Detection

Motion detection is an important part of the system. It provides the system with a recovering mechanism if the rest of the system for some reason<sup>5</sup> should fail to pursue a target.

Motion detection functions continuously during pursuit and is used as an integral part in selecting points where the target is lying. The threshold for detecting independently moving points is on purpose set in a non-committing way. This results in many points detected outside of the moving target(s), but what is important is that we get the points on the target.

---

<sup>4</sup>This is a term used in personal communication with Kristian Simsarian and we make no claim to be originators.

<sup>5</sup>In real-world situations this will happen sooner or later.

The effects of this will be clear in the experiments. The motion detection module is shown schematically in Figure 2.

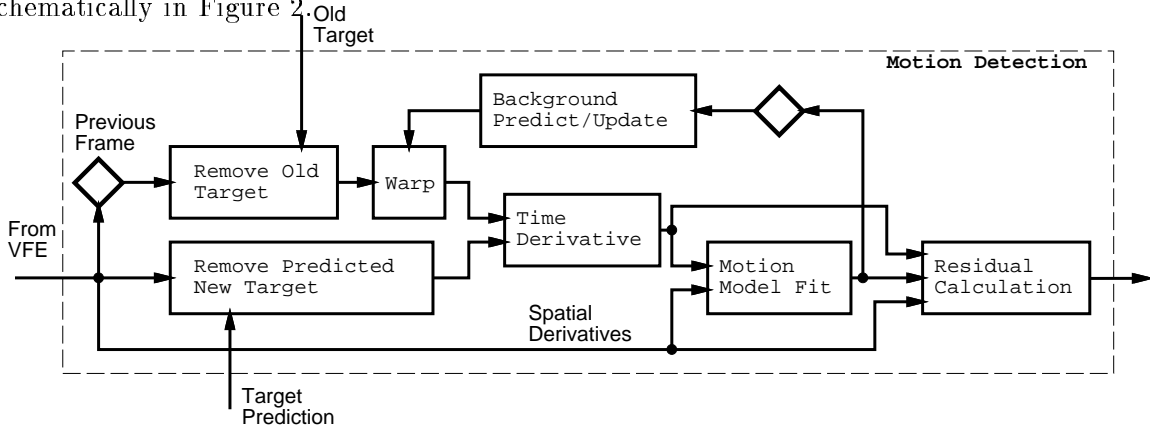


Figure 2: The motion detection is shown schematically. This is a detailed description of the **Motion Detection** box in Figure 1. (The diamonds indicate a one frame delay in the feedback.)

**Affine Background Motion Model** Using the brightness constancy constraint a global affine image velocity model is fit to the gradient and time derivative information (see **Motion Model Fit** in Figure 2 and Appendix A.1). Two steps involving feedback are included to account for object motion and large background motion.

- The predicted and previous position and extent of the target is used to mask out parts of the image which likely belong to the object, so that they do not affect the calculation of the affine parameters for the background. See feedback into **Motion Detection** in Figures 1 and 2. This is a crucial step to avoid iterations that otherwise would have been necessary to get reasonable performance even if targets are large. We could say that the iterations are performed in time instead of in a single frame.
- The accumulated affine parameters are used over time to cancel out the majority of the time difference, see feedback into **Warp** in Figure 2. Thereby we can avoid iterations in fitting the affine model, and instead iteratively refine our motion model in time. This is very important to increase the computational speed and decrease the subsequent lag of the pursuit mechanism. We have here an example of how one can tradeoff the performance of a single algorithm for better overall performance.

**Background Segmentation** Once the fit is made to the background, and the affine parameters are available, the residual normal image velocity is calculated in the entire image, **Residual Calculation** in Figure 2 (see Appendix A.1.2). The obtained residual is thresholded and high residuals are considered as possible independently moving areas in the scene. The threshold on the residual is adaptive and is set relative to the difference between target and background motion,  $T_b = \max(T_b^0, F_b |\mathbf{v}_t - \mathbf{v}_b|)$ , where  $\mathbf{v}_t$  and  $\mathbf{v}_b$  are the target and background translation respectively,  $T_b^0$  is the minimal threshold accepted, and  $F_b$  a discriminability factor. The result of this can be seen in the presentation of the experiments (see Section 6.3 and Figure 8).

### 4.1.2 Target Segmentation

Target segmentation is an important complement to the motion detection mechanism and the disparity segmentation. The aim is to determine which parts of the scene are moving consistently with what is currently believed to be the target. This is the typical situation in connection with occlusions, when the target gradually disappears, to later emerge on the other side of the occlusion. It also supplies the system with the ability to discriminate between several moving targets, found by the motion detection, and distinguish parts of the scene that lie at the same depth, found by the disparity segmentation. This is shown schematically in Figure 3.

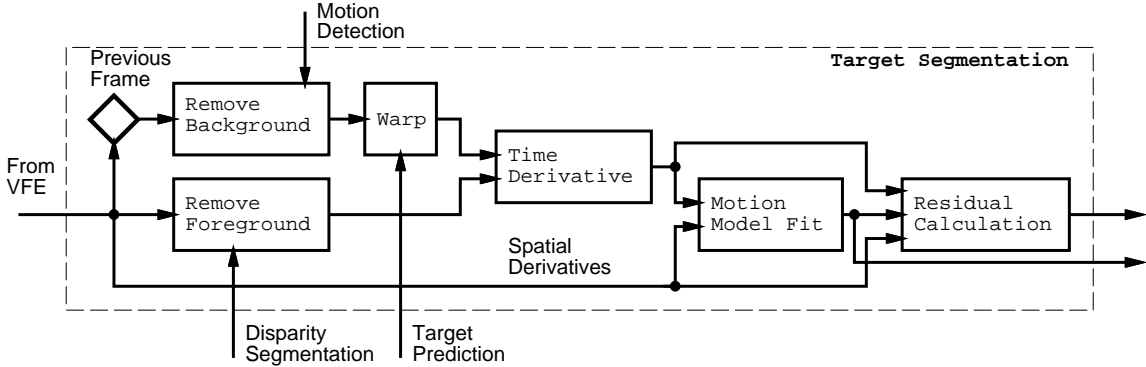


Figure 3: The target segmentation is shown schematically. This is a detailed description of the **Target Segmentation** box in Figure 1. (The diamonds indicate a one frame delay in the feedback.)

**Affine Target Motion Model** The calculations on the target are performed in analogy with what is done for the background motion, with the advantage that most of the computations spent on the background can be used also here (spatial derivatives and smoothed images from the VFE). By sharing these data we greatly increase the performance. One difference is worth noting here, the target is much smaller than the background, resulting in fewer points that can be used in the fitting of the model. This calls for tools to accumulate information of the model over time, and not just in one frame. Here we could use some standard technique, filtering the affine parameters with some weighting scheme. However this brings us far from the actual data and it becomes hard to define a reasonable weighting scheme and to relate errors between the measurements and the model. We have instead chosen another approach.

In the affine fit, sums like  $\sum I_x I_y y$  are computed over the area which is to be fitted (see Appendix A.1, Equation 9). It is natural to average these sums over time since they are all done over the same target (or background). We have implemented the averaging as  $\hat{S}_{(t)} = \alpha S_{(t)} + (1 - \alpha)\hat{S}_{(t-1)}$ , where  $S_{(t)}$  is one of the newly computed sums, and  $\hat{S}_{(t-1)}$  the corresponding previously used sum. This produces a new sum,  $\hat{S}_{(t)}$ , that is used to compute the affine parameters. The averaging is natural in the sense that it forgets over time (the target will not always move the same way), and that as the target area degrades, more information is used from the past (conversely if the target area becomes bigger).

The effects of this averaging are however not conclusive. In some cases it is very useful e.g. when there is a shrinkage of the area in the image which is covered by the target, while in other cases it has the opposite effect, of keeping old data that are not correct.

**Target Segmentation** Analogous to the motion detection (see Section 4.1.1), the residual normal image velocity is calculated and an adaptive thresholding is applied,  $T_t = \max(T_t^0, F_t | \mathbf{v}_t - \mathbf{v}_b |)$ ,  $T_t^0$  is the minimal threshold accepted, and  $F_t$  a discriminability factor. Here low residuals are considered as parts of the target. The result of this can be seen in the presentation of the experiments (see Figure 9).

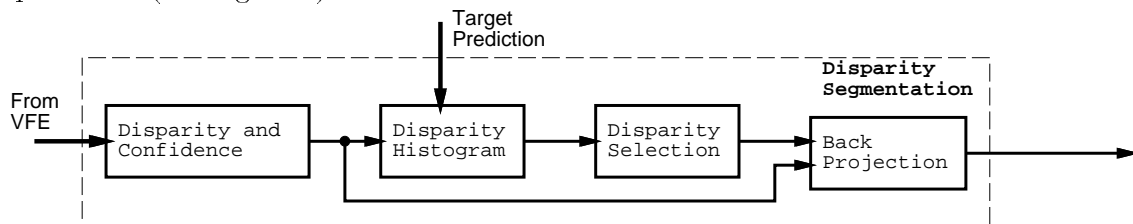


Figure 4: The disparity segmentation is shown schematically. This is a detailed description of the **Disparity Segmentation** box in Figure 1.

### 4.1.3 Disparity Segmentation

To increase performance of the system it has also proven useful to have relative depth information about locations in the scene, if areas are closer than the target or behind the target. Disparity segmentation has two principal parts, the computation of disparities locally and a selection process to find the disparities corresponding to the target, in order to separate the target from the background and foreground. This is shown schematically in Figure 4.

**Disparity and Confidence Computation** The disparity computation is carried out with the phase based method presented in Appendix A.2, and a confidence measure provides a way to disregard the unreliable disparity estimates. The phase based method has been chosen because of its low complexity, and because it fits well into the VFE framework, which means that we can use the same data for vergence and pursuit movements. It also performs well in a real-time implementation for vergence on the KTH-Head-Eye system, see Section 5.1.

**Disparity Selection** The object of disparity selection is to select the disparities that belong to the target in the presence of disparities that arise from other locations in the scene. In Maki *et al.* (1994) and Maki *et al.* (1993) some early work on disparity selection was presented. This work has been further developed to take more advantage of the feedback obtained when the system is run in a closed loop together with other cues to follow a single target, to ensure that version and vergence components in smooth pursuit remains consistent. Also other researchers have presented work on disparity selection, e.g. Taylor *et al.* (1994).

The basis for selection is the predicted location and extent of the target for which a dominant disparity,  $D_t$ , is chosen. This is done by selecting the highest peak in a histogram (see **Disparity Histogram** in Figure 4) of the disparities that lie within the target prediction (a mask similar to the ones in Figure 7). A range of disparities around  $D_t$  are said to belong to the target.

## 4.2 Integration

We will in this section in more detail describe how the different components of the system are integrated and how they benefit from each other. The integration is performed mainly

in **Target Update**, see Figure 1, but in the **Target Segmentation**, Figures 1 and 3, motion detection information and disparities are also integrated.

The target extent is kept in a mask. The integration to produce an updated target mask uses masks produced in Sections 4.1.1 and 4.1.2, such as the ones shown in Figures 8 and 9. Pixels that are black are considered as possible target pixels. There are two parts needed to update the target mask, one to exclude pixels and one to include pixels.

**Target Exclusion** Target areas that do not get support from either motion detection or target segmentation are excluded from the target model. Also the disparity module detects areas in the scene that lie in front of the pursued target, which are then excluded from the target model. One could argue that also background areas could be removed using disparity, but then the problem of occlusion arises; scene parts that lie beyond the target will be occluded by the target which makes disparity detection impossible in these areas. It is exactly in the areas around the target, where disparity is not available, that such information is valuable. A disparity algorithm that detects also occluded areas would be helpful here, but we have not developed any real-time implementation of this so far.

**Target Inclusion** Areas in the scene that are detected as both moving independently, as detected by motion detection, and are moving consistent with the target image velocity model from the target segmentation, are added to the target model. That is, if pixels arise in both Figure 8 and 9, they are considered as target pixels.

## 5 Real-Time Implementation

Presently we have implemented some parts of what has been described above to perform real-time visual processing coupled to motor control of the KTH-Head-Eye system. A full scale real-time implementation is however in progress and our real-time implementations so far lead us to believe that we will complete this shortly. We will briefly describe two parts that have been implemented, and some visual results are presented in the experimental section.

### 5.1 The Real-Time Pursuit Mechanism

Let us first describe a simplified version of the above system running in real-time. It includes not only all the parts from above, motion detection, target velocity and disparity estimation, but also motor control of the KTH-Head-Eye system, implementing smooth pursuit, vergence and saccades in a closed loop. The implementation is made on a T800 Transputer network, where both the motor control and visual processing takes place.

The system works in two modes, while during one mode the smooth pursuit is being performed, the other mode detects luminance changes in the images for a subsequent saccade to this location. The switching between the two takes place automatically when, (1) the target stops during smooth pursuit, at which time the change detection starts, and (2) when the saccade has finished after a change detection, the smooth pursuit begins. Figure 1 shows also this real-time system, if **Background Motion Model** and **Target Update** are excluded. Also no connection exists from **Disparity Segmentation** and **Target Detection**.

**Image Velocity Estimation for Smooth Pursuit** The velocity estimation is done in the center of a cyclopean image using a binomial pyramid with four layers, in which the velocity

estimation is derived coarse to fine. The cyclopean image is constructed by simply averaging between the left and right image.

After capturing the left and right images the two images are averaged in the central region ( $9 \times 9$ ) to give the cyclopean image. The cyclopean image is then convolved with a separable binomial filter (order five corresponding to  $\sigma = 1.0$ ) and subsampled by two in each direction. This is repeated for the resulting image two more times, giving a pyramid with a total of four images ( $93 \times 93$ ,  $45 \times 45$ ,  $21 \times 21$ ,  $9 \times 9$ ).

Starting with the coarsest level ( $9 \times 9$ ), the central patch ( $7 \times 7$ ) in the old pyramid together with the central patch ( $7 \times 7$ ) in the new pyramid yields a sum of squared differences, SSD. This is repeated for the eight neighboring patches (displaced by one pixel) in the top level of the new pyramid. The position of the smallest SSD value is propagated down to the next level in the pyramid ( $21 \times 21$ ), and the same calculation of the nine SSD ( $13 \times 13$ ) is performed at this level but now centered around the location given by the coarser level. Repeating this for the last two levels with  $25 \times 25$  and  $49 \times 49$  windows gives the final estimate of the motion between the last two frames.

The maximum measured motion is 15 pixels between frames in both the horizontal and vertical direction, and it operates in 25 Hz with about 50 ms delay from the occurrence in the world. The delay includes 20 ms image waiting (we need two frames), and 30 ms for computation. It should be noted here that the 20 ms of waiting is not processor idle time, but computation on the previous frame pair is being performed during this time.

**Disparity Estimation using a Log-Polar mapping** In the most integrated version of the pursuit system, a log-polar mapping has been used to give disparity estimation for vergence control, Capurro and Uhlin(1994). One novel result of this integration was the robustness of the control system even in the presence of converging and diverging cameras. Another promising result is that the integration is possible with the current control system although the disparity estimates are given at 5 Hz, while the pursuit signal computed from image velocity is measured, and the pursuit controlled, 25 times per second.

For sample experiments see Section 6.2.

**Phase Based Disparity Estimation** Maki *et al.* (1993) have demonstrated the use of phase based disparity estimation for vergence control on our Head-Eye system. Vergence errors were measured with this phase based disparity estimator at 25 Hz to control the cameras to continuously and smoothly verge correctly on an approaching moving target. Even though this was not fully integrated with version movements<sup>6</sup> we believe that it will function well in a real-time integration. Further support for this is the real-time integration with disparity estimation using the log-polar mapping, which in the current implementation gives much worse estimation and at a slower rate.

**Frame Differencing for Target Selection** For motion detection the real-time system uses frame differencing. This is done separately on the left and right images. The differencing is performed on images that subsampled by 16 in each dimension after averaging with an  $16 \times 16$  box-filter. The largest change between two such images in time is selected out of the left and right difference, and if this exceeds a threshold, a saccade is performed. This process runs at about 10 Hz, thus a saccade is triggered with about a 150 ms delay.

---

<sup>6</sup>Current hardware limitations stopped us from performing a full integration.

## 5.2 The Real-Time Motion Detection Mechanism

A real-time implementation of a more complex algorithm for detecting moving targets in the presence of camera and observer ego-motion, similar to what is presented in Section 4.1.1, has also been made. This is integrated together with the control of the KTH-Head-Eye system, controlling the gaze-direction, enabling pursuit of an independently moving target although the platform and the head are continuously and independently moving.

**Motion Detection for Target Selection** The real-time implementation of the motion detection algorithm is in its current experimental setup used to visually servo our robot head to follow a small moving target. This system runs continuously, capturing image data directly from one camera on the robot head, and in a closed loop controls the camera motion to track the moving object. The image processing takes place on a MaxVideo200 (a pipeline processing machine). The robot head is controlled by a T800 Transputer network. We here model the image velocity somewhat simpler than an affine image transformation. The image velocities  $\hat{\mathbf{v}}$  (see Appendix A.1) are modelled as purely translational.

**Implementation details and algorithm** All intermediate calculations are performed with 16-bit integer precision. The captured images are smoothed before computing central differences and time derivatives. A shift is performed using the predicted background speed, before performing the time difference, this corresponds to the **Warp** in Figure 2. This figure shows the this implementation if the two target inputs are excluded.

When the difference image is computed it is truncated to 8-bit precision and histogrammed. The histogram is used to decide a threshold. The threshold is set to keep a certain number of imagepoints when producing a binary image<sup>7</sup>. The centroid  $C$  of the binary image is computed.

The motor control continuously receives  $C$  as the target location in the image and controls both pan and tilt<sup>8</sup> to visually center the target. For an experiment see Section 6.7. The control is exactly the same as in the real-time experiments with smooth pursuit.

The system calculates the time-derivative with a time-difference of 1/25 second between 2 consecutive frames. The centroid computation and camera control runs at a rate of 6.5 Hz, using an image of the size  $300 \times 250$  pixels<sup>9</sup>, and smoothing of the images with  $\sigma = 5$ <sup>10</sup>.

## 6 Experiments and Results

In this section we will show how the system performs in a number of different situations, both in the presence of all cues and when some cues are not present. The target will in most examples undergo complex movement, including rotations and movements towards and from the camera, while the observer undergoes ego-motion. No knowledge about any of these motions is assumed.

During most of the experiments the images used were recorded during real-time pursuit performed by the already existing pursuit mechanism on the Head-Eye system, see Section 5.1. Therefore the experiments are performed on quite realistic data since all the noise due to

---

<sup>7</sup>To truncate to 8-bit precision, and later only use a binary image is a tradeoff for computational speed.

<sup>8</sup>The motor control runs on the Transputer network and involves both eye and neck movements, see Pahlavan *et al.* (1992).

<sup>9</sup>This gives a  $25.7^\circ$  viewing angle in the horizontal direction, and  $21.5^\circ$  viewing angle in the vertical direction.

<sup>10</sup>A  $5 \times 5$  binomial filter is cascaded 5 times

inaccurate control of the head is incorporated, while motion blur, out of focus blur and vergence errors are also present. All in all the image sequences are captured in 25 Hz, during camera motion which is purely image driven with no human interference (except that there are humans walking in front of the cameras, being pursued). This facility enables us to experiment on a variety of sequences under different conditions exemplifying the performance of the system presented in Section 4 in many different situations.

## 6.1 Experimental Setup

All experiments were performed with the head-eye-system constructed by Pahlavan (1993). This head-eye-system has now been mounted on a mobile platform, see Figure 5.

Images have been grabbed at a frame rate of 25 Hz. Images from both left and right cameras are grabbed synchronously, using 2 MaxVideo200s as frame-grabbers. From the cameras<sup>11</sup> we only use 1 interlace which gives us images of the size  $366 \times 287$  pixels, and a shutter speed of  $1/50$  sec. The lenses used are zoom lenses<sup>12</sup>, set to the widest viewing angle: 657 pixels focal length, or  $31^\circ$  and  $24.6^\circ$  viewing angle in the horizontal and vertical direction respectively.



Figure 5: The experimental platform is a Head-Eye-System mounted on a mobile platform.

**General Experimental Considerations**<sup>13</sup> Each image in the sequences are 8 bit integer images as they come directly from the digitizer. Throughout the experiments these images ( $366 \times 287$ ) are smoothed with a  $9 \times 9$  kernel ( $\sigma = 2$ ), and then a  $3 \times 3$  binomial filter is used before subsampling by 2. The subsampling produces a floating point 32 bit image and all the subsequent calculations are carried out with the same precision on the resulting  $183 \times 143$

<sup>11</sup>The robot head is equipped with 2 CCIR cameras, Philips LDH 670.

<sup>12</sup>Camera lenses:Ernitec, model M12Z6. Manufacturers specifications: Focal length 12.5-75 mm, max aperture: F1.2, horizontal angle of view for 2/3" chip:  $6^\circ 7' - 38^\circ 7'$ , (this differs a bit from what we practically can use).

<sup>13</sup>This applies to the experiments in Sections 6.3 through 6.5

images. Central differences are used to produce spatial derivatives in the subsampled images. The difference between two consecutive frames are used as time derivatives<sup>14</sup>.

The residual normal velocity images, see Sections 4.1.1 and 4.1.2, are smoothed with a  $9 \times 9$  kernel ( $\sigma = 2$ ), before thresholding. In the disparity calculation the resulting disparities are smoothed in the same way, though weighted by their confidence values.

The automatic thresholding between background and target, see Sections 4.1.1 and 4.1.2, is set to  $1/3$  ( $= F_b = F_t$ ) of the translational velocity difference between target and background motion. The minimal value for this threshold is set to 1.0 ( $= T_b^0 = T_t^0$ ) for both target and motion detection.

The parameter settings for the disparity segmentation have not been fully automated, the threshold for what is considered closer than the target is set to  $D_t + 1.0$  pixels<sup>15</sup> (see Section 4.1.3). The number of layers used in the disparity pyramid is also fixed to 5 and the confidence threshold is 20.0.

The maximum target size is set to  $128 \times 128$ .

**Performance** At the moment we lack hardware for trying the full system in real-time. However, there exists an implementation of the system with all its functionalities, running at 9 seconds per cycle<sup>16</sup>, including all the computations involved in the binocular pursuit and motion detection. Running the system monocularly, thus removing disparity computations and the motion computations for one camera, the cycle time is 2.5 seconds. The timing was performed on a SUN Sparc10. For the performance of the real-time implementations refer to Section 5 and respective experiments below.

## 6.2 Real-Time Pursuit with Motor Control

The performance of the real-time pursuit mechanism is shown in Figures 6<sup>17</sup>, 10 and 13 (see Section 5.1). We can see how the pursuit mechanism centers a target visually while the target is moving across the room. The visual processing and the control are performed in 25 Hz and these captured sequences are used in the subsequent three experiments to show the increased performance achieved with the system described in Section 4, especially in conjunction with occlusions. In Figure 6, when the target moves behind the occluding object in frame 22, the pursuit does not follow the target across to the other side, but stays on the occluding object.

## 6.3 Handling Occlusions

As was said in the previous experiment the real-time pursuit system, which does not explicitly handle occlusions, does not follow the target past the occluding object in Figure 6.

With figure ground segmentation provided by the system described in Section 4, the system can handle partial occlusions and segment out the target, see Figure 7. We can see how the occluding pole is not included in the points representing the target as it passes behind. We describe in more detail what is happening (the numbering below refers to the frame numbers in Figure 7).

When the moving target is found (2) the system will start to incorporate areas which it believes to belong to the target (3). As the target is picked up the system will e.g. exclude

---

<sup>14</sup>IIR filtering has been tested but it seems to degrade system performance, especially when there are occlusions.

<sup>15</sup> $D_t$  changes in time as the target moves in depth.

<sup>16</sup>The timing includes all the preprocessing, of beginning with the raw left and right images.

<sup>17</sup>For an explanation of the overlay we refer to the next section.



Figure 6: A sequence captured during real-time pursuit. The pursuit system computes an image velocity on every frame (25 per second), to pursue the target. Shown here are every 6<sup>th</sup> frame recorded.

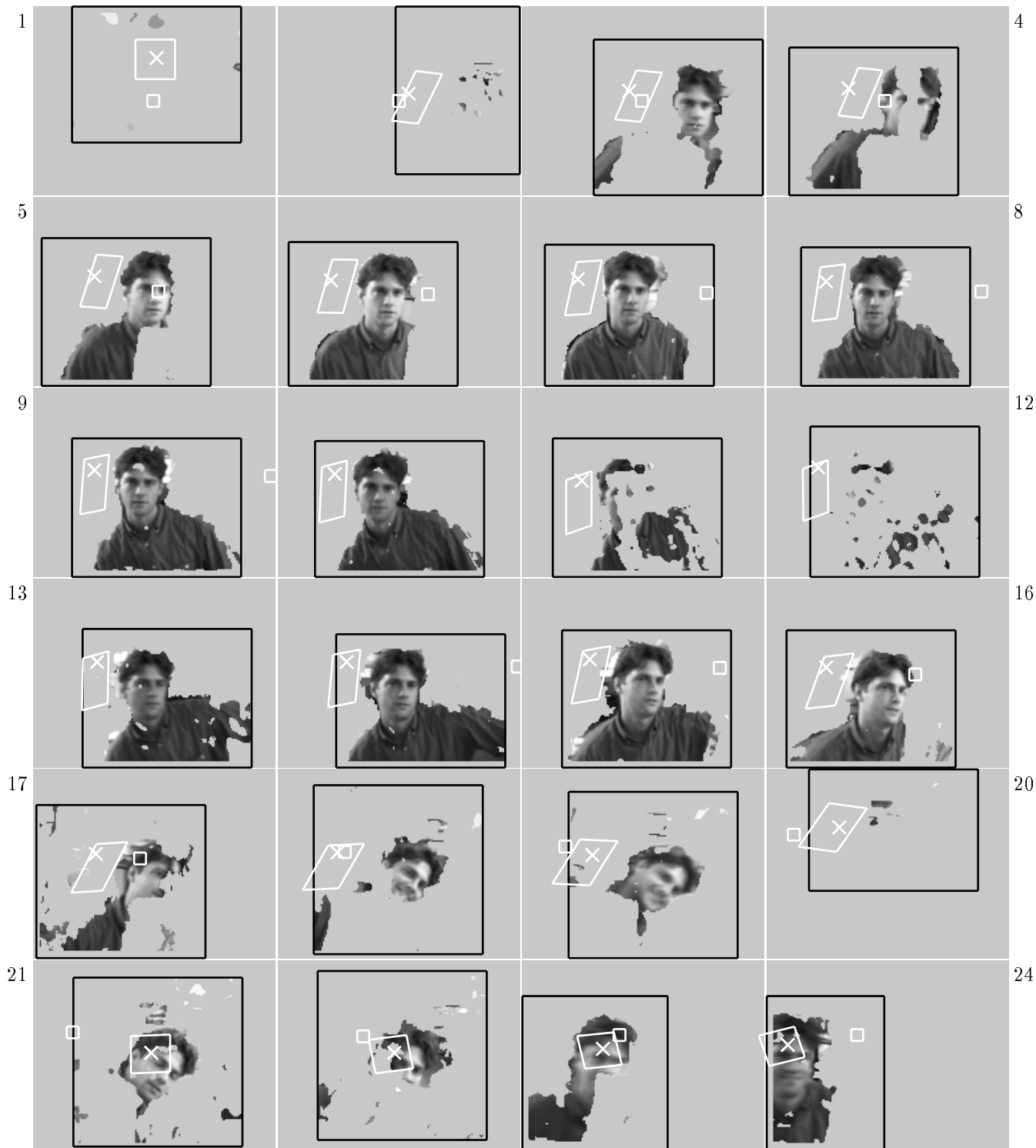


Figure 7: Target pixels as extracted by the system from the sequence shown in Figure 6.



Figure 8: Motion detection returns areas that possibly belong to a moving target. Shown here are the areas which the motion detection marks as possible target pixels. These are the motion detection masks used to produce the final target masks, (compare with odd frames in Figure 7).

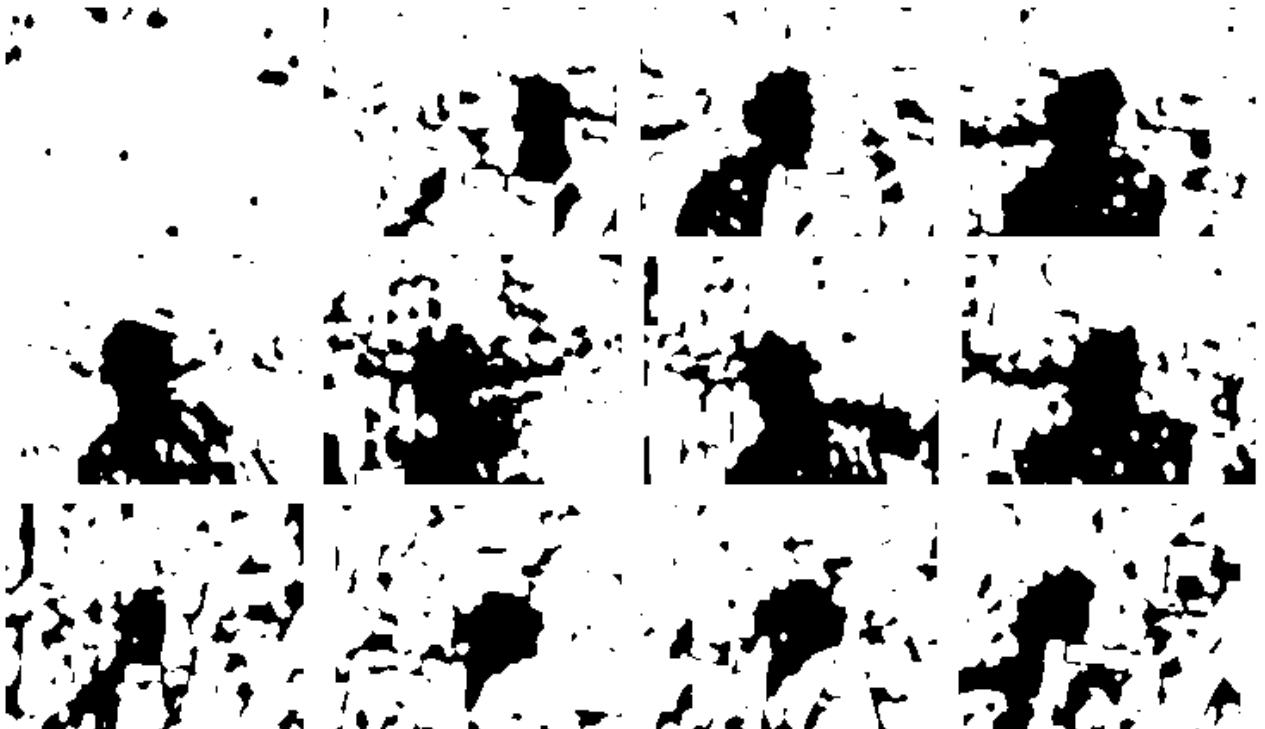


Figure 9: Target segmentation returns areas that are believed to belong to the pursued target. Shown here are the areas which are consistent with the target image velocity model. These are the target segmentation masks used to produce the final target masks, (compare with odd frames in Figure 7).

areas that are occluding since the occluding object moves together with the background, rather than the target (4). As the target comes out behind the occluding object new areas are incorporated in the target until the full upper body appears (8). As the target moves to the left it gradually stops moving relative to the background and subsequently the motion information disappears (12). The target is picked up again as it starts moving in the other direction and again the system picks up the whole upper body as a target (13-16). As the target becomes occluded, the occluded regions disappear from the target, to be picked up again as the head appears on the other side. Again the target stops and the motion information is lost (20), but as the target starts moving the head is once again picked out. Now we can see the benefit from this processing: while the target was lost by the simplified real-time pursuit process, it is now correctly pursued past the occluding object.

The result of the affine fit to the target is shown in Figure 7 as a rectangle which is allowed to distort accordingly. An interesting detail in this experiment is the qualitatively correct rotation of the rectangle towards the end of the sequence, where the upper body and head rotates clockwise and as the target moves back to the left, the rectangle turns counter clockwise. It is also here that we notice the largest difference in having an affine model compared to a purely translational one. When we use a purely translational model the target is not pursued past the occluding object since the area consistent with this model will not be large enough, and the target is lost.

We would like to stress that the absolute location and shape of this distorted rectangle is of no relevance. The absolute location is determined when a moving target is detected and its subsequent movement is driven by the affine fit to the target image velocity (see Appendix A.1). This means that early on, just after detection the location of the rectangle moves around quite randomly until a large enough area of the target is picked up to stabilize the movement. What is of importance though is the rectangle's relative movement between frames, that reflects target movement, lateral, vertical, rotational and expansion, all of which are important descriptors in the pursuit of a target.

The small rectangle shows a fixed point on the background as calculated during background cancellation in motion detection. Notice especially that this point indeed remains stationary relative to the background (see Figure 6). Looking into the first and last images shown in the figure we see that the slip is about 1 pixel in the horizontal direction and 3 pixels in the vertical direction over a total of 139 frames<sup>18</sup>.

The white cross shows the result of coarse-to-fine correlation when performed with the target masks produced by the system.

The large black rectangle shows a window that is automatically placed around the centroid of the target pixels which are flagged as belonging to the target in each frame. Only pixels inside this rectangle are kept to the next frame. It is put there to restrict the target and get rid of some spurious areas that might come up far from the target. It is a simplified way to achieve some sort of closeness of pixels belonging to the target. An alternative would be to use some sort of connected components, but this is ill suited for real-time implementation since it has no support on our current hardware. The effect of this rectangle is little in this particular experiment, the only difference is that a few more spurious pixels are detected in some frames. It is kept anyway to get the same conditions in all experiments.

To achieve this result the system uses two masks, one produced by motion detection, Figure 8 (see Section 4.1.1), and another from target detection Figure 9 (see Section 4.1.2).

---

<sup>18</sup>A maximum angular speed of 20 degrees per second was reached during this experiment.

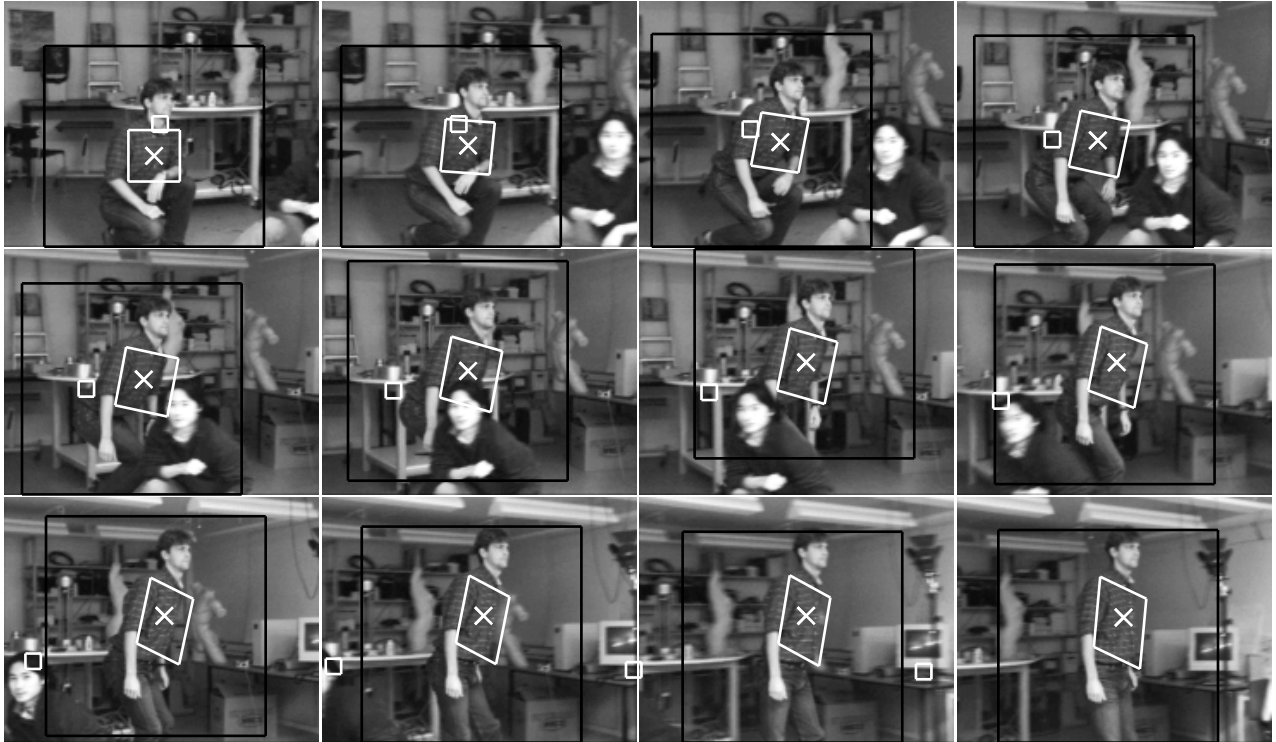


Figure 10: A sequence captured during pursuit. The pursuit is performed at 25 Hz, and shown here are every 6<sup>th</sup> frame.

## 6.4 Occlusion and Other Moving Objects

To show also how the system performs in the presence of other moving targets, we have performed experiments when another object not only moves in the scene, but also partially occludes the pursued target as it moves across the scene, see Figure 11.

When an occluding object is present, and it also is moving, it will be included by motion detection as a possible target location. If this object in some way dominates, it may well take over the attention of the system if it relies only on motion. To show that, we have removed the disparity detection of areas in front of the target with the result shown in Figure 12. The attention is shifted to the person moving in front, although the attention was initially on the person moving behind. Even though this may be an unwanted behavior of the system, it anyway shows that the system can stably change its attention.

To conclude, we see that without the disparity cue, the system is easily distracted by moving occluding targets, while with the disparity cue, successful pursuit is achieved even in the presence of such distractors.

## 6.5 Expanding Targets

The ability to detect and pursue an object that moves toward the observer is of great importance since such an object presents a potential collision. It is then important to receive a measure which is directly linked to the approach of the target so that an avoidance maneuver can be deployed in time. Although just qualitatively shown here, the affine model provides such a measure, which is depicted by the distorted white rectangle in Figure 13, together with the result of the target pursuit and segmentation. This measure is however not used explicitly here, but to conclude, we see that pursuit is functioning also when the target is approaching.

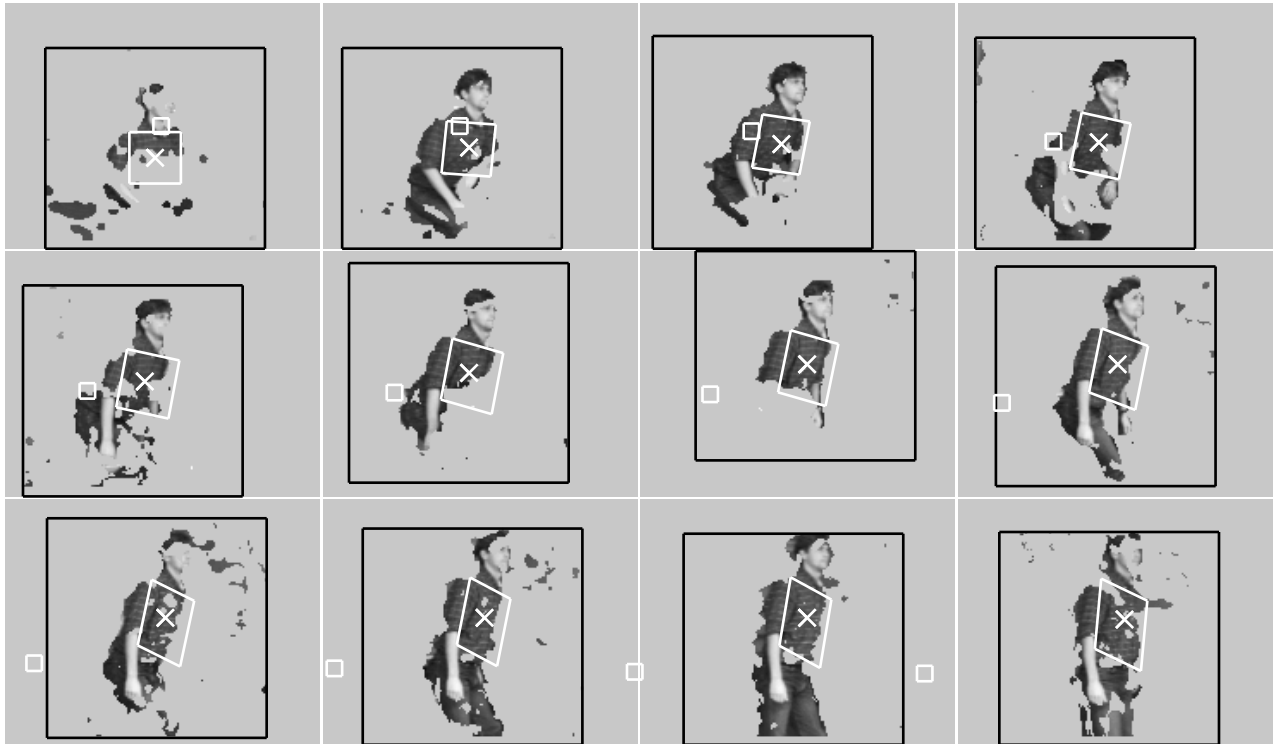


Figure 11: Target pixels as extracted by the system from the sequence shown in Figure 10. Every 6<sup>th</sup> frame is shown here.

## 6.6 Rotating Targets

The performance during target rotation is shown in Figure 14, where the segmentation is shown with black overlay. First of all it is seen that the affine model does pick up the rotation of the target and that a correct pursuit is performed. Not as many points as in the previous experiments are picked up on the target in the segmentation, mainly because of the uniform areas on the umbrella, in which no local image velocity information is available. Another important factor here is the thresholding employed, although it is adjusted automatically by the system, it is global. A lower threshold should be employed in the lower part of the umbrella where the motion difference between target and background is smaller.

While all the previous experiments were performed with the real-time pursuit mechanism as the base, this experiment is performed in a manual way. The reason is that the current real-time implementation does not handle rotating targets very well, making it difficult to obtain a prolonged pursuit for recording. This sequence was recorded at 25 Hz while the cameras were panning slowly to the right. The panning was performed manually and thus no

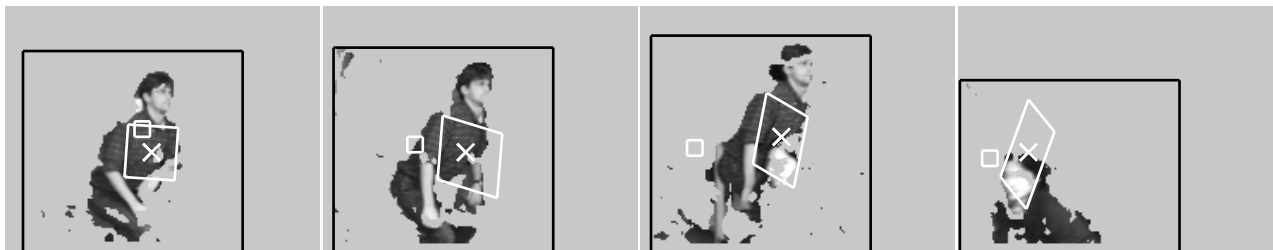


Figure 12: Target pixels as extracted by the system from the sequence shown in Figure 10, but without the disparity cue. The attention shifts to the second moving person. (Every 12<sup>th</sup> frame is shown here. To be compared with images 2, 4, 6 and 8 in Figure 11.)

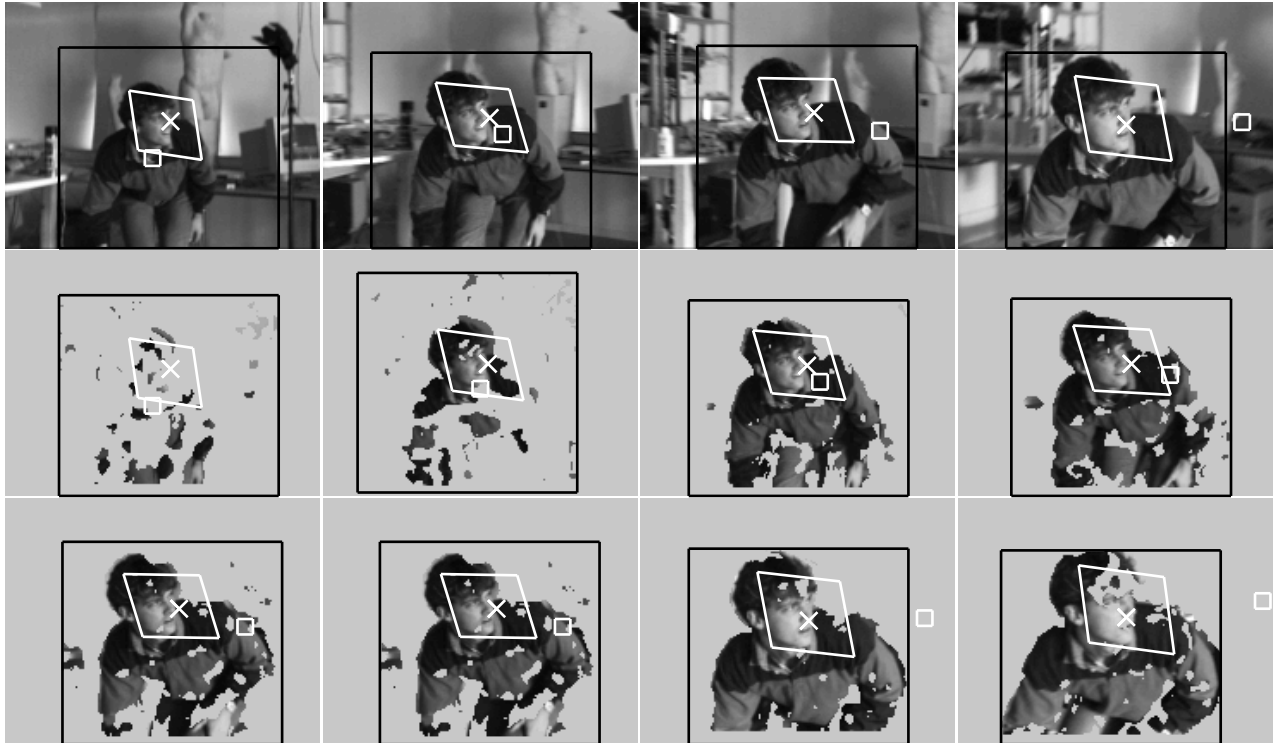


Figure 13: Pursuit during target expansion. On the top is shown the original sequence recorded during real-time pursuit, where every 6<sup>th</sup> frame is shown. In the bottom row is shown the result of the segmentation every 3<sup>rd</sup> frame.

real-time visual feedback occurs during this experiment.

## 6.7 Real-Time Motion Detection with Motor Control

In the experiment seen in Figure 15 the algorithm described in Section 5.2 runs with visual input from one camera only. The Figure shows images taken every 2 seconds from a video-record of the experiment which is performed in real-time with visual feedback to the motor control.

The system is tracking the white paper-box<sup>19</sup> moving diagonally through the room from left to right, seen from the camera. As the camera is tracking the box, the platform is moving approximately straight ahead, at frame 14 in Figure 15 the platform slows down, stops and starts to move straight backwards. In frame 13 to 20 the camera is not tracking as good as in the beginning. The reason for this is that the motion of the box is close to the maximum of what this system currently handles<sup>20</sup>. Notice that the system is capable of tracking the object both in the presence of a rather smooth untextured background (frames 1-5), and a quite cluttered background (frames 6-19).

During this experiment the platform moves a distance of 10 meters forward in 28 sec, (frames 1-14). After the stop the platform moves about 7 meters backwards in 18 sec, (frames

<sup>19</sup>The white paper box is hanging in a steel wire attached to the end of a stick. A person is moving this box through the room.

<sup>20</sup>This implementation is a feasibility study in which the motor control has not been tuned for the delays introduced by the processing time.

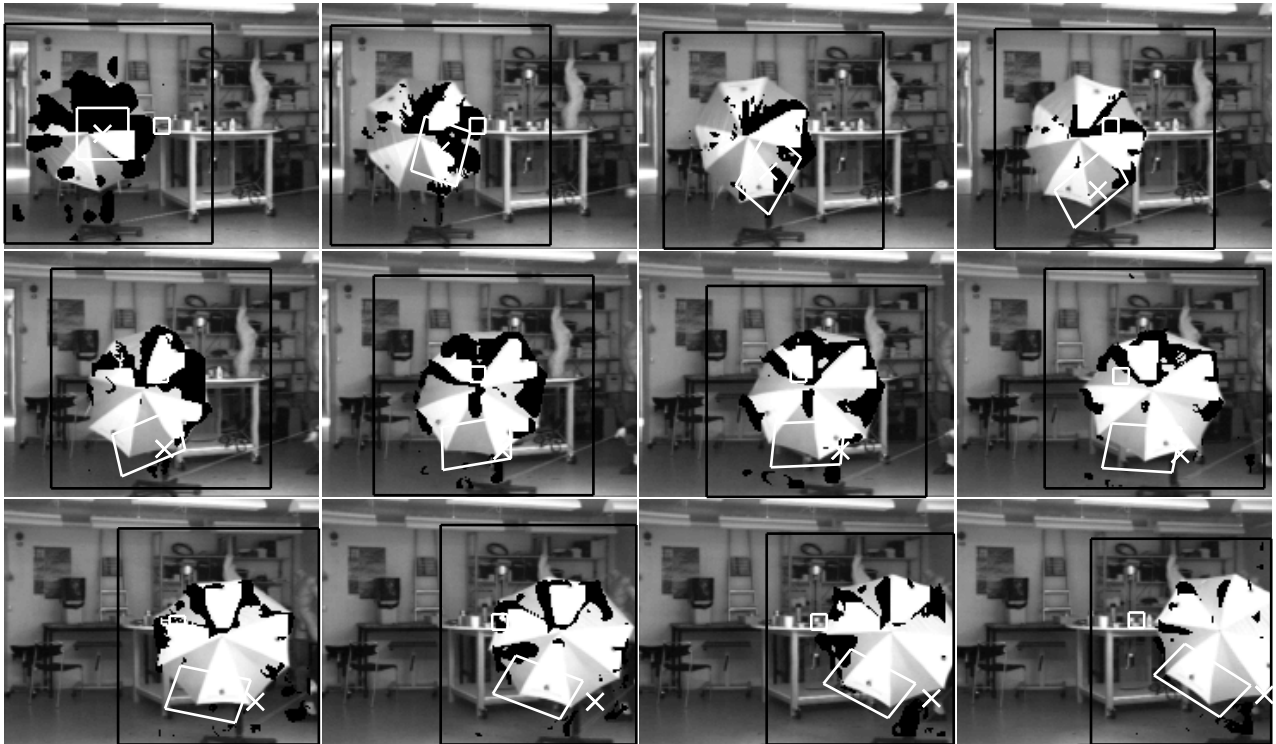


Figure 14: Pursuit of a rotating umbrella. The sequence shows every 6<sup>th</sup> frame in a 25 Hz sequence. The black areas show which areas are used for the affine fit depicted by the distorted white rectangle.



Figure 15: The figure shows the motion detection mechanism with real-time motor-control feedback, tracking the white box. The algorithm is running at 6.5 Hz (for details about the experiment see Section 6.7). The interval is 2 seconds between each frame shown in the figure.

16-24). The background motion is approximately 6 pixels/frame. This yields an angular velocity of  $6.5^\circ/\text{sec}$ .

The performance of this system would be greatly improved if the images were subsampled before the motion detection algorithm is being run, since this would allow a much higher processing rate, 25 Hz instead of 6.5 Hz, and thus cause less delay in the control of the system. Moreover, a subsampling would make it possible to cope with higher angular velocities of the camera. This has been shown in the off line experiments, where the motion detection functions very well over  $20^\circ/\text{sec}$  angular velocity.

## 7 Discussion

In this paper we have argued that to develop active "seeing" systems we by considering the problem from a systems perspective gain notable advantages. This perspective in particular includes three key components.

- Continuous operation over time and real-time response to different types of events.
- An open and expandable design such that the system can grow while data and results from all levels are shared.
- The integrated use of multiple cues, which is shown to help solving the figure-ground problem in complex, non-static situations.

The presented work forms part of a development of an active visual observer, that can "see things" while moving about in the environment. Here we demonstrate the abilities to attend to a target that moves and to maintain fixation on such a target. In particular we have shown that these basic abilities, if implemented in real-time and using different cues, can be quite robust in varying situations and that in fact they provide solutions to the figure-ground problem. Hence, this work can be claimed as providing important steps towards our goal.

The emphasis on the systems approach and real-time performance has led us to consider simple algorithms. Hence, we do not claim to any particular contributions to any of the partial problems on deriving information from a certain cue. However, we argue that robust performance can be obtained even with simple methods, given an integrated approach of the proposed type. Our experiments support this claim.

Of course there remains much work to be done to achieve our goals. First, although we stress real-time operation we still lack appropriate high bandwidth interfacing hardware to realize that. Currently, we are for instance not able to use our VFE together with the coarse-grained multiprocessor layer, which prevents us from running intermediate level algorithms on 3-D structure, while holding gaze. This missing link will soon be added. Another remaining question concerns the assessment of our results. So far we have performed experiments with different characteristic events and motion patterns. However, work on how we can measure both observer and object motion objectively in these experiments is under way. In line with our argument that it is essential to utilize that "the world is rich", we believe that such measurements will provide a better understanding of the problem than pure simulations. Ongoing work will show if we are correct in these claims.

## A Theory

In this section we will briefly go through the theory that lies behind the algorithms used in Section 4.

### A.1 Brightness Constancy and Affine Image Velocity

#### A.1.1 Brightness Constancy

Using the notation  $I_x = \frac{\partial I(\mathbf{x}, t)}{\partial x}$ ,  $I_y = \frac{\partial I(\mathbf{x}, t)}{\partial y}$ ,  $I_t = \frac{\partial I(\mathbf{x}, t)}{\partial t}$  and  $\nabla I(\mathbf{x}, t) = (I_x(\mathbf{x}, t), I_y(\mathbf{x}, t))^T$ , the brightness constancy can be written as,

$$\nabla I(\mathbf{x}, t) \cdot \mathbf{v}(\mathbf{x}, t) + I_t(\mathbf{x}, t) = 0 \quad (1)$$

where  $\mathbf{v} = (\frac{dx}{dt}, \frac{dy}{dt})^T$  is the image velocity. This equation is not enough to constrain the two parameters of  $\mathbf{v}$ , given the gradients ( $\nabla I$ ), and the time derivatives ( $I_t$ ). What can be determined though, is  $\mathbf{v}$ 's component normal to the gradient, the normal image velocity.

#### A.1.2 Residual Normal Velocity Calculation

From equation (1) we can determine the normal component of the velocity vector locally as:

$$v_n(\mathbf{x}) = -\frac{I_t(\mathbf{x})}{|\nabla I(\mathbf{x})|}. \quad (2)$$

With a arbitrary velocity field,  $\hat{\mathbf{v}}(\mathbf{x})$ , and the gradients in an image, we can write the normal velocity as:

$$\hat{v}_n(\mathbf{x}) = \frac{\nabla I(\mathbf{x}) \cdot \hat{\mathbf{v}}(\mathbf{x})}{|\nabla I(\mathbf{x})|}, \quad (3)$$

and define the residual between this and the observed normal velocity as:

$$R(\mathbf{x}) = \hat{v}_n(\mathbf{x}) - v_n(\mathbf{x}) = \frac{\nabla I(\mathbf{x}) \cdot \hat{\mathbf{v}}(\mathbf{x}) + I_t(\mathbf{x})}{|\nabla I(\mathbf{x})|} \quad (4)$$

#### A.1.3 Solving for the Affine Image Velocity

To solve for a velocity field we must use a model for the same, and if we let  $\hat{\mathbf{v}}$  be parameterized with  $\mathbf{u}$ , giving  $\hat{\mathbf{v}} = \hat{\mathbf{v}}(\mathbf{u}, \mathbf{x})$ , we can pose a weighted least squares minimization problem to solve for  $\mathbf{u}$ ,

$$\min_{\mathbf{u}} = \sum_{\mathbf{x} \in \Omega} w(\mathbf{x}) R(\mathbf{u}, \mathbf{x})^2 \quad (5)$$

where  $\Omega$  is a region of interest in the image where the parameterized velocity model should hold.

**The Affine Velocity Model** We model the image velocity,  $\hat{\mathbf{v}}$  as one affine motion for an image region  $\Omega$ . The number of parameters are then 6 which yields,

$$\hat{\mathbf{v}}(\mathbf{u}, \mathbf{x}) = B(\mathbf{x})\mathbf{u} = \begin{bmatrix} a + bx + cy \\ d + ex + fy \end{bmatrix} \quad (6)$$

$$\mathbf{u} = (a, b, c, d, e, f)^T, \quad B(\mathbf{x}) = \begin{bmatrix} 1 & x & y & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & x & y \end{bmatrix}$$

where  $a, b, c, d, e, f$  are unknown scalar constants.

**Solving for the Affine Motion Parameters** The weighting function  $w$  in the minimization in Equation (5) is chosen as the gradient magnitude squared, i.e.  $w(\mathbf{x}) = \nabla I(\mathbf{x})^2$ . Then we have the following minimization problem,

$$\min_{\mathbf{u}} \sum_{\mathbf{x} \in \Omega} (\nabla I(\mathbf{x}) \cdot \hat{\mathbf{v}}(\mathbf{u}, \mathbf{x}) + I_t(\mathbf{x}))^2. \quad (7)$$

By using equations (6) and (7), a region  $\Omega = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ , and measurements of the gradients and time derivatives in the image, we get the following linear equation system,

$$V^T V \mathbf{u} = V^T \mathbf{q} \quad (8)$$

where

$$\mathbf{V} = \begin{bmatrix} I_x(\mathbf{x}_1) & I_x(\mathbf{x}_1)x_1 & I_x(\mathbf{x}_1)y_1 & I_y(\mathbf{x}_1) & I_y(\mathbf{x}_1)x_1 & I_y(\mathbf{x}_1)y_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ I_x(\mathbf{x}_n) & I_x(\mathbf{x}_n)x_n & I_x(\mathbf{x}_n)y_n & I_y(\mathbf{x}_n) & I_y(\mathbf{x}_n)x_n & I_y(\mathbf{x}_n)y_n \end{bmatrix} \quad \text{and} \quad \mathbf{q} = \begin{bmatrix} -I_t(\mathbf{x}_1) \\ \vdots \\ -I_t(\mathbf{x}_n) \end{bmatrix}$$

Which is a  $6 \times 6$  symmetric positive semi-definite system, with the 6 elements of  $\mathbf{u}$  as unknowns, shown explicitly in equation (9). It becomes definite as soon as there are more than one gradient direction present in the region over which the minimization is performed.

$$\begin{bmatrix} \sum I_x^2 & \sum I_x^2 x & \sum I_x^2 y & \sum I_x I_y & \sum I_x I_y x & \sum I_x I_y y \\ & \sum I_x^2 x^2 & \sum I_x^2 xy & \sum I_x I_y x & \sum I_x I_y x^2 & \sum I_x I_y xy \\ & & \sum I_x^2 y^2 & \sum I_x I_y y & \sum I_x I_y xy & \sum I_x I_y y^2 \\ & & & \sum I_y^2 & \sum I_y^2 x & \sum I_y^2 y \\ & & & & \sum I_y^2 x^2 & \sum I_y^2 xy \\ & & & & & \sum I_y^2 y^2 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} = - \begin{bmatrix} \sum I_t I_x \\ \sum I_t I_x x \\ \sum I_t I_x y \\ \sum I_t I_y \\ \sum I_t I_y x \\ \sum I_t I_y y \end{bmatrix} \quad (9)$$

*symmetric*

With all the sums performed over the region to which we want to fit the model.

## A.2 Phase-Based Disparity Estimation

The fundamental concept of the phase-based approach to disparity estimation is to recover the local disparity as the spatial shift from the local phase difference observed in the Fourier domain. The phase is extracted locally in the left and the right stereo images by taking the argument of the convolution product with a complex filter, such as a Gabor filter. As the local shift between stereo images is approximately proportional to the local phase difference, a disparity estimate at each point in the image is derived accordingly. Extensive work have been carried out in phase-based disparity estimation, e.g. Sanger(1988), Wilson and Knutsson(1989), and for details on the techniques employed here, see Maki *et al.* (1993).

**Disparity Map** Let a complex filter be convolved with the left and right images respectively and produce  $V_l(\mathbf{x})$  and  $V_r(\mathbf{x})$  at each point  $\mathbf{x}$  in the image. Since the computation of the convolution in the Fourier domain has contribution mainly from the neighborhood of the central filter frequency  $\omega$ [radian/pixel], the convolution products are approximately related to each other by a phase shift according to the frequency,  $\omega \cdot D(\mathbf{x})$ , where  $D(\mathbf{x})$  denotes disparity. The relation therefore leads to the disparity through the computation of the complex phase difference:

$$D(\mathbf{x}) \approx (\arg V_l - \arg V_r)/\omega. \quad (10)$$

This is strictly valid only for filters of infinitesimal bandwidth, arising directly from the Fourier shift theorem, Sanger(1988).

**Confidence Map** In order to check the feasibility of the estimated disparity and threshold unreliable estimation, we also compute a confidence value defined on the basis of the magnitude of the convolution product:

$$C(\mathbf{x}) \equiv \frac{|V_l| |V_r|}{|V_l| + |V_r|}. \quad (11)$$

**Complex Filter** While the complex filter has to satisfy several constraints and several different filters have been proposed accordingly, we employ discrete approximations to the first and second derivatives because of their computational simplicity, that is  $(-1, 0, 1)$  and  $(1, 0, -2, 0, 1)$ , thus sacrificing accuracy. The frequency of the filter is regarded as  $\omega = \pi/2$ [radian/pixel]. The maximum disparity which a filter can accurately determine is limited to one-half the wavelength of the filter. To handle larger disparities, a coarse-to-fine technique is employed.

## References

- Andersen, C. S. and Christensen, H. I. (1994). Using multiple cues for controlling an agile camera head. In *Proc. of the Workshop on Visual Behaviours*, pp. 97–101, Seattle, Washington.
- Ballard, D.H. (1991). Animate vision. *J. of Artificial Intelligence*, 48, 57–86.
- Capurro, C. and Uhlin, T. (1994). Vergence and pursuit using a log-polar mapping. (in preparation).
- Coombs, D. and Brown, C. (1992). Real-time smooth pursuit tracking for a moving binocular robot. In *Proceedings 1992 IEEE Comp. Soc. Conf. on Computer Vision and Pattern Recognition*, pp. 23–28, Champaign, IL.
- Dickmanns, E. D., nad C. Brüdigam, R. Behringer, Dickmanns, D., Thomanek, F., and v. Holt, V. (1993). An all-transputer visual autobahn-autopilot/copilot. In *Proc. 4th International Conference on Computer Vision*, pp. 608–615.
- Gårding, J. and Lindeberg, T. (1995). Direct computation of shape cues based on scale-adapted spatial derivative operators. *International Journal of Computer Vision*. (To appear).
- Grimson, W. E. L., Ratan, A. Lakshmi, O'Donnell, P. A., and Klanderma, G. (1994). An active visual attention system to play “where’s waldo”. In *Proc. of the Workshop on Visual Behaviours*, pp. 85–90, Seattle, Washington.
- Irani, M. and Peleg, S. (1993). Motion analysis for image enhancement: Resolution, occlusion, and transparency. *Journal of Visual Communication and Image Representation*, 4, no. 4, 324–335.
- Koenderink, J. J. and van Doorn, A. J. (1987). Representation of local geometry in visual system. *Biological Cybernetics*, 55, 367–375.
- Maki, A., Uhlin, T., and Eklundh, J.-O. (1993). Phase-based disparity estimation in binocular tracking. In K. A. Høgdra, B. Braathen, K. Heia, editor, *Proc. 8th Scandinavian Conference on Image Analysis*, pp. 1145–1152, Tromsø, Norway. Norwegian Society for Image Processing and Pattern Recognition.
- Maki, A., Uhlin, T., and Eklundh, J.-O. (1994). Disparity selection in binocular pursuit. In *Proc. 4th IAPR (International Association for Pattern Recognition) Workshop on Machine Vision Applications*, pp. 182–185.
- Murray, D. W., P. F. McLaughlin, I. D. Read, and Sharkey, P. M. (1993). Reactions to peripheral image motion using a head/eye platform. In *Proc. 4th International Conference on Computer Vision*, pp. 403–411.
- Pahlavan, K. (1993). *Active Robot Vision and Primary Ocular Processes*. Ph. D. dissertation, Dept. of Numerical Analysis and Computing Science, KTH (Royal Institute of Technology). ISBN KTH/NA/P--93/16--SE.
- Pahlavan, K., Uhlin, T., and Eklundh, J.-O. (1992). Integrating primary ocular processes. In Sandini, G., editor, *Proc. 2nd European Conference on Computer Vision*, volume 588 of *Lecture Notes in Computer Science*, pp. 526–541, Santa Margherita Ligure, Italy. Springer Verlag, New York.
- Pahlavan, K., Uhlin, T., and Eklundh, J.-O. (1993). Dynamic fixation. In *Proc. 4th International Conference on Computer Vision*, pp. 412–419, Berlin, Germany. IEEE Computer Society Press.
- Prokopowicz, P. N., Swain, M. J., and Kahn, R. E. (1994). Task and environment-sensitive tracking. In *Proc. of the Workshop on Visual Behaviours*, pp. 73–78, Seattle, Washington.
- Sanger, T. D. (1988). Stereo disparity computation using gabor filters. *Biological Cybernetics*,

59, 405–418.

- Sundareswaran, V., Bouthemy, P., and Chaumette, F. (1994). Active camera self-orientation using dynamic image parameters. In Eklundh, J.-O., editor, *Proc. 3rd European Conference on Computer Vision*, volume II, pp. 111–116, Stockholm, Sweden.
- Taylor, J., Olson, T., and Martin, W. N. (1994). Accurate vergence control in complex scenes. In *Proc. IEEE Comp. Soc. Conf. on Computer Vision and Pattern Recognition, 1994*, pp. 540–545, Seattle, Washington.
- Thorpe, C., Herbert, M., Kanade, T., and Shafer, S. (1992). *The New Generation System for the CMU Navlab*. Springer Verlag.
- Tölg, S. (1992). Gaze control for an active camera system by modeling human pursuit eye movements. In *SPIE Intelligent Robots and Computer Vision IX*, volume 1825, pp. 585–598.
- Wang, John Y. A. and Adelson, Edward H. (1994). Spatio-temporal segmentation of video data. In *Proc. of the SPIE: Image and Video Processing II*, volume 2182, San Jose.
- Wilson, R. and Knutsson, H. (1989). A multiresolution stereopsis algorithm based on the Gabor representation. In *Proc. IEEE International Conf. Im. Proc. and Applic.*, pp. 19–22, Warwick. U.K.