Combination of Simple Vision Modules for Robust Real–Time Motion Tracking

Joachim Denzler and Heinrich Niemann

denzler,niemann@informatik.uni-erlangen.de

The following paper will be published in the

European Transactions on Telecommunications
Vol.6, Nr. 3, 1995

Combination of Simple Vision Modules for Robust Real–Time Motion Tracking

Joachim Denzler, Heinrich Niemann
Universität Erlangen–Nürnberg
Lehrstuhl für Mustererkennung (Informatik 5)
Martensstr. 3, D–91058 Erlangen
Tel.: +49–9131–85-7894, FAX: +49–9131–303811
email: {denzler,niemann}@informatik.uni-erlangen.de

Abstract

In this paper we describe a real time object tracking system consisting of three modules (motion detection, object tracking, robot control), each working with a moderate accuracy, implemented in parallel on a workstation cluster, and therefore operating fast without any specialized hardware. The robustness and quality of the system is achieved by a combination of these vision modules with an additional attention module which recognizes errors during the tracking.

For object tracking in image sequences we apply the method of active contour models (snakes) which can be used for contour description and extraction as well. We show how the snake is initialized automatically by the motion detection module, explain the tracking module, and demonstrate the detection of errors during the tracking by the attention module.

Experiments show that this approach allows a robust real–time object tracking over long image sequences. Using a formal error measurement presented in this paper it will be shown that the moving object is in the center of the image in 90 percent of all images.

1 Introduction

In computer vision the field of real time image processing has become more and more important. Because of the increasing computation performance – a rule of thumb says that every two years the computational power doubles – real time image processing can now be applied to real world applications. Autonomous vehicles, service robots, or machines assisting handicapped persons are expected to be constructed within the next ten years. One important task for such machines is real time motion detection and real time object tracking. In traffic scenes one has to detect and track other moving objects, or service robots cleaning the floor have to avoid stationary and moving obstacles, like humans or animals.

Recently so called *data highways* were expanded allowing to transmit a great amount of data in high speed. Therefore, in near future in telecommunication more and more applications transmitting images will come up. Also the importance of image processing will rise.

For example, one wishes to talk via teleconferencing to each other as close as possible to a face–to–face conversation, i.e. not rigidly sitting in front of the camera, and even being fully in the middle of the camera image. In such an application the camera's position has to be changed that the two partners always are kept in the middle of the camera image. Of course, the tracking of the speaker's head has to be done in real time.

For real time object tracking it is necessary to have a closed loop between sensors – which produces the image data – and action – for example moving the camera to track a moving object. Moving the camera implies that new image data will be transferred into the system and therefore new actions have to be performed. If a moving system detects an obstacle, actions have to be carried out to avoid them, and the calculations necessary to achieve this task have to be done in real time. For object tracking the system has to compute the necessary motion of its camera fast enough to keep the object in the field of view. From these requests it follows that not necessarily 25 images per second (the normal video rate) have to be processed but that an upper bound for the processing time has to be met in which the tracking task can be performed. Violating this upper bound results, for example, in losing the moving object. We will call this upper bound the *in time constraint*, which is an important characterization of real time systems [13].

So far for real time image processing specialized hardware, and special parallel computers had to be used, for example, for preprocessing, filtering, or calculation of optical flow. The disadvantage of such an approach is that with a new generation of more powerful hardware the algorithms have to be reimplemented and new software using the new hardware has to be designed. This costly programming overhead can be avoided if a real time image processing system is implemented in a standard programming language running on arbitrary workstations. The difficulty is that todays general purpose workstations lack the required transmission rate for real time image processing given a frame rate of 25 images per second. To process each 512×512 frame of a sequence of color images taken at video rate – 25 images per second – 20 Million bytes per second have to be handled. On a 120 MIPS workstation only 6 instructions for each pixel could be used. Assuming a load/store architecture 2 instructions are needed to load the pixel and to store the result. So with the remaining 4 instructions a global task like image processing would have to be done. This is unrealistic even if there were 40 instructions available. Some authors estimate that one needs up to 10^4 instructions per pixel to get a complete high level interpretation of an image [11, 17].

To handle the amount of data in real time image processing one has to change from a more static processing strategy to a purposive one, i.e. not every pixel of the image should be processed but only the interesting parts to perform a given task. This purposive processing strategy is summarized in a new paradigm called *active vision* which came up in the field of computer vision [2, 5, 8, 24]. The main principle of active vision can be described by a definition of Aloimonos [3]:

Active Vision: *Given a problem of computer vision: Take the images in a way that the problem can be solved*

Different from that the approach of Marr [19] says:

Marr: *Given images: Design algorithms to solve a problem in the field of computer vision with these images*

Some of the main technical mechanisms of active vision are pyramids, attention modules, or – in general – selectivity in space, time and resolution [24].

During the past five years many authors have proven that using the active vision paradigm real time constraints can be satisfied, and promising algorithms and methods have been developed. One of these methods – especially for object tracking – is the so called *active contour model* also referred to as *snake* [15]. A modified and improved version of snakes is used and described in this paper.

In our problem domain a toy train moving in front of a robot has to be tracked. The robot has a camera mounted on its hand. By moving the camera according to the tracking results the toy train is always supposed to be in the middle of the camera image. As an additional constraint no specialized hardware for preprocessing, filtering, or segmentation will be used. All algorithms are implemented on standard $UNIX^{©}$ workstations in an object–oriented programming language and environment [10, 23].

The remainder of this paper is organized as follows: In the next section we will introduce the principles of active contour models and motivate the use of active contours for object tracking. In Sect. 3 the concepts of vision modules for real time object tracking are described. Then, a complete object tracking system built up of such vision modules is introduced (Sect. 4). We will present an automatic initialization of snakes, and an error detection based on features extracted out of the snake itself. After this we will introduce our experimental environment. Experiments will show that this system is both robust and fast enough to track a moving toy train in front of a robot in a closed loop of action and vision. The paper ends with a summary and discussion of the results and gives an outlook on further work that has to be done in this context.

2 Snakes: Active Contour Models

The energy minimizing model of active contours was first introduced by Kass [15]. An active contour is an energy minimizing spline which is influenced by its own internal energy E_{int} and by external forces E_{ext}. A snake $\mathcal{S}$ of n discrete contour points can be defined as a parametric function $\boldsymbol{v}(s)$

$$\boldsymbol{v}(s) = (x(s), y(s)),\ ; s \in 0, \cdots, n-1\ , x(s) \in [0, x_{max}], \quad y(s) \in [0, y_{max}] \tag{1}$$

where x_{max} and y_{max} are usually given by the size of the input image. Such an active contour has an energy E defined by

$$E \ = \ \sum_{s=0}^{n-1} \left(E_{int}(\boldsymbol{v}(s)) + E_{ext}(\boldsymbol{v}(s))\right). \tag{2}$$

E_{int} is usually defined as (cf. [15])

$$E_{int}(\boldsymbol{v}(s)) = \frac{\alpha(s)|\boldsymbol{v}_s(s)|^2 + \beta(s)|\boldsymbol{v}_{ss}(s)|^2}{2}, \tag{3}$$

and $\boldsymbol{v}_s(s)$ and $\boldsymbol{v}_{ss}(s)$ are the first and second derivatives of $\boldsymbol{v}(s)$. The external forces may be the image intensity $f(x, y)$, or the edge strength of an image, for example $E_{ext}(\boldsymbol{v}(s)) = -|\nabla f(\boldsymbol{v}(s))|^2$. Using the edge strength as the external force during the energy minimization the snake will be pushed to strong edges, for example, to the contour of an object.

For energy minimization many approaches exist in the literature, for example, Kalman filtering [25], dynamic programming [4], finite element methods [9, 14] and a variation of the model based on the normal optical flow [7]. One way described in [15] is based on the variational calculus and will be briefly summarized in the following.

Searching for a function $\boldsymbol{v}(s)$, which minimizes equation 2, leads to the *Euler-Lagrange differential equation.* In the discrete case the following equations must be solved [15]:

$$\boldsymbol{A}\boldsymbol{x} + \boldsymbol{f}_x(\boldsymbol{x}, \boldsymbol{y}) = 0 \tag{4}$$

$$\boldsymbol{A}\boldsymbol{y} + \boldsymbol{f}_y(\boldsymbol{x}, \boldsymbol{y}) = 0 \tag{5}$$

with

$$\boldsymbol{x} = (x(0), x(1), \ldots, x(n-1)), \boldsymbol{y} = (y(0), y(1), \ldots, y(n-1)),$$

$$\boldsymbol{f}_x(\boldsymbol{x}, \boldsymbol{y}) = \left(\left.\frac{\partial E_{ext}}{\partial x(s)}\right|_{s=0}, \left.\frac{\partial E_{ext}}{\partial x(s)}\right|_{s=1}, \ldots, \left.\frac{\partial E_{ext}}{\partial x(s)}\right|_{s=(n-1)} \right),$$

and

$$\boldsymbol{f}_y(\boldsymbol{x}, \boldsymbol{y}) = \left(\left.\frac{\partial E_{ext}}{\partial y(s)}\right|_{s=0}, \left.\frac{\partial E_{ext}}{\partial y(s)}\right|_{s=1}, \ldots, \left.\frac{\partial E_{ext}}{\partial y(s)}\right|_{s=(n-1)} \right).$$

The matrix $\boldsymbol{A}$ contains the influence of the internal energy of the snake, i.e. the parameters α and β (for a complete derivation see [15]). For the computation of the unknown vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ an iterative procedure is used that converges if $\boldsymbol{x}_t = \boldsymbol{x}_{t-1}$, where $\boldsymbol{x}_t$ is the solution at step t. Thus, equation 4 and 5 can be written as:

$$\boldsymbol{A}\boldsymbol{x}_t + \boldsymbol{f}_x(\boldsymbol{x}_{t-1}, \boldsymbol{y}_{t-1}) = 0 = \gamma(\boldsymbol{x}_{t-1} - \boldsymbol{x}_t) \tag{6}$$

$$\boldsymbol{A}\boldsymbol{y}_t + \boldsymbol{f}_y(\boldsymbol{x}_{t-1}, \boldsymbol{y}_{t-1}) = 0 = \gamma(\boldsymbol{y}_{t-1} - \boldsymbol{y}_t) \tag{7}$$

with γ being the stepsize, and transformed to:

$$\boldsymbol{x}_t = (\boldsymbol{A} + \gamma\boldsymbol{I})^{-1}(\gamma\boldsymbol{x}_{t-1} - \boldsymbol{f}_x(\boldsymbol{x}_{t-1}, \boldsymbol{y}_{t-1})) \tag{8}$$

$$\boldsymbol{y}_t = (\boldsymbol{A} + \gamma\boldsymbol{I})^{-1}(\gamma\boldsymbol{y}_{t-1} - \boldsymbol{f}_y(\boldsymbol{x}_{t-1}, \boldsymbol{y}_{t-1})) \tag{9}$$

Because of the special form of $\boldsymbol{A}$ (penta–diagonal) [15] $(\boldsymbol{A} + \gamma\boldsymbol{I})$ can be inverted efficiently by an LU decomposition of complexity $O(n)$ [6].

The principle of the active contour models is clarified in Figure 1; one can see a snake with seven snake elements positioned on an energy field. For example this energy field could be computed from the negative edge strength of a circle using a standard edge operator, e.g. a Sobel operator. Now, during the energy minimization step each snake element slithers downhill to the next minimum. In this case the snake elements will rest at the edge extracting the contour of the circle. Further information concerning the snake model and its behavior can be found in [15] and [18].

In several papers, for example [7, 18], the advantages of snakes for object tracking were shown. Given an image sequence $f_0(x, y), f_1(x, y), \ldots, f_n(x, y)$ containing a single moving

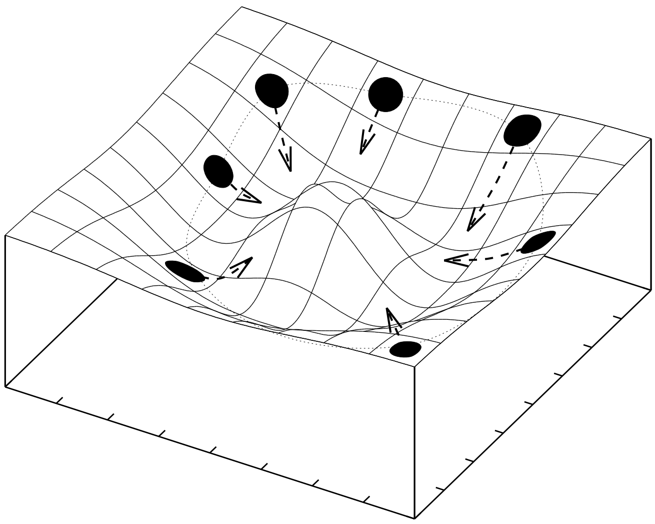

Figure 1: Principle of active contours: A snake with 7 elements extracts the contour of a circle by moving into the minimum of the negative edge strength of the circle.

object it is only necessary to initialize the active contour on the contour of the moving object within the first image. Then the contour of the moving object can be tracked by placing the snake $\boldsymbol{v}_t(s)$ of image $f_t(x, y)$ on the image $f_{t+1}(x + \triangle x, y + \triangle y)$ where $(\triangle x, \triangle y)$ is the estimated velocity of the contour in the 2D image plane. Setting $(\triangle x, \triangle y) = (0, 0)$ means that no prediction of the position of the contour in the subsequent image is done. Then, if the object is moving sufficiently slow in comparison to the elapsed time between f_t and f_{t+1}, the snake will extract the object's contour in the image $f_{t+1}(x, y)$ by energy minimization (see also Figure 4 in Sect. 4.4).

Compared to other feature matching algorithms for real time object tracking active contours provide several advantages:

- Computation of the external energy E_{ext} (see equation 2) demands processing of only small parts of an image, namely the region around the snake elements.
- Object extraction and object tracking is done within one step.
- No object/background distinction is needed. This is especially advantageous if the camera is moving, too.
- If the moving object is partially occluded the snake is able to retain the object's contour for a while [25].

- Features of a contour (for example the center of gravity) are more stable than single point features. This makes the tracking more robust.

There are also some disadvantages. First the parameters α, β, and γ have to be chosen — in most of the cases heuristically. Second, the snake must be positioned close to the contour which should be tracked. This is often done in an interactive way. In Sect. 4.3 we present a method for an automatic placement of the snake near a moving object in an image sequence grabbed with a static camera.

Active contour models without prediction, i.e. $(\triangle x, \triangle y) = (0, 0)$, are very well suited for real time tracking problems where a homogeneous background exists, the tracked object has a smooth contour, and the contour of the object can also be extracted from a low resolution image. In the literature an extension to rigid snakes can be found [7]. Suited parameter selection (α and β in equation 3) may allow the snake to form a corner but it is difficult to choose these values automatically. Experiments showed that a prediction step is essential for tracking in a natural environment, i.e. with a background containing strong edges from other objects as well.

3 Vision Modules for Object Tracking

In [1] the utilization of vision modules is specified. Various simple modules each performing a single and specific task communicate with each other. By integration of the results from all vision modules a more complicated vision task like object tracking may be executed. Before giving an example for a real time tracking system built up of such vision modules in Sect. 4 we will look at the general problem of object tracking.

For object tracking in a closed loop between sensors and action the following modules will be used in our work (see Figure 2):

- a motion detection module,
- a tracking module,
- a module for the robot control, and
- an attention module for detection of certain events during tracking.

It should be noted that in our approach the attention module has not only the purpose of detecting interesting parts in the image but also to supervise the whole system, for example to detect errors, and to control the cooperation between the modules. Consequently one should extend the general pattern analysis system (see [21], page 12) to the requirements of an active vision system which means error detection, and focusing on interesting parts of the image.

Some other modules could be added, for example, for knowledge based object tracking, or for the classification of the tracked object [10]. Then, if the object is classified, it is possible to track specific features of that object (model based tracking), or to check the correctness of the data driven tracking after fixed intervals (model based error detection).

There are some advantages and, of course, some disadvantages of such an approach. The main disadvantage is the moderate accuracy and reliability of each vision module working on its own. Thus, errors will occur, and a mechanism has to be provided to detect such errors.

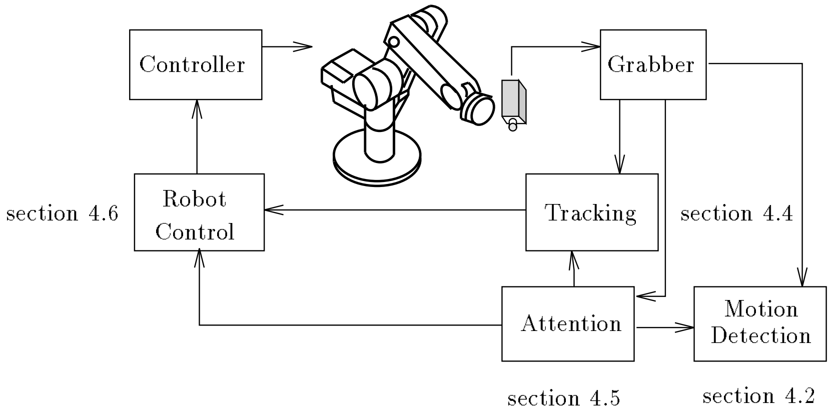

Figure 2: The principle of vision modules for object tracking

Therefore, the overall system performance is given by the attention module and its ability to detect and fix errors. Our experiments will show that a robust system can be built using an error detecting attention module. Then the advantage is that because of the simplicity of the modules the *in time constraints* can be satisfied.

Single modules working in parallel will allow to map the modules to a workstation cluster. Using the `pvm`-library [12] a virtual parallel machine can be built where each workstation performs the dedicated subtask of the vision module. The mapping is done by the `pvm` transparently for the user. Otherwise the `pvm` allows for running all vision modules on a single workstation. Of course, this will cause a small decrease of the computation power due to communication overhead in contrast to a system running on a single workstation without `pvm`.

At this implementation stage we use only two separate processes in our system. The robot control is one of the processes, the attention module, motion detection and tracking is the second one. The reason will be discussed in Sect. 4 where we present an example of a vision system based on the ideas of vision modules.

4 Robust Tracking of a Moving Object with a Combination of Simple Vision Modules

4.1 General Idea

The goal is to keep a moving object in the middle of the camera image. To build up a system of vision modules we have to decompose the problem. First, the motion in the scene has to be detected and a region of interest has to be localized in which motion occurs (Sect. 4.2). For that purpose the whole view of the camera at a low resolution of 128×128 pixels is used.

Then we have to initialize the snake on the object's contour. Actually, most of the authors used an interactive initialization of the snake, and we do not know of any automatic initialization published yet which is suitable for real time applications. We present a fast and robust method for an automatic initialization which is described in Sect. 4.3.

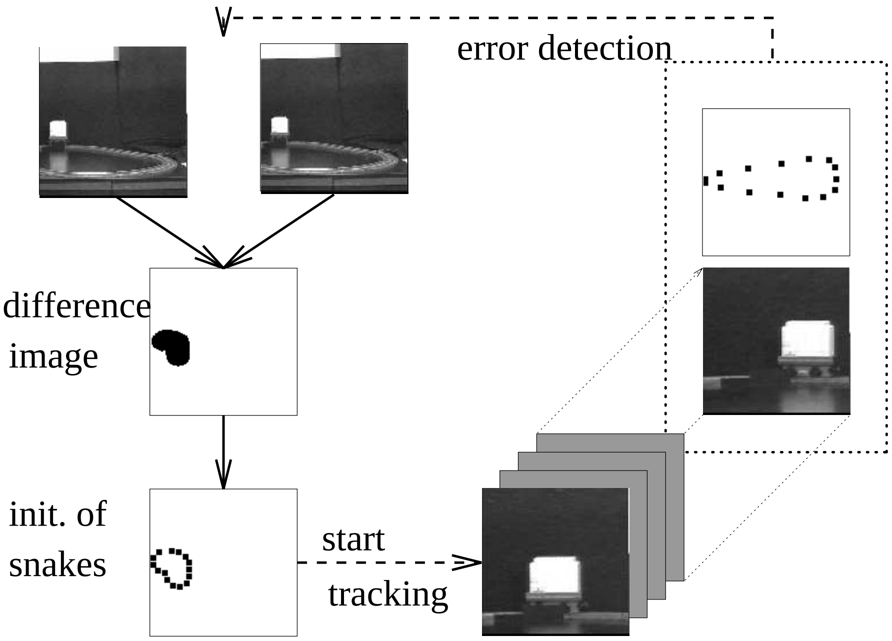

Figure 3: Overview of the motion detection and tracking modules. The black region is the ROI as a result of the difference image between two images. Sampling the chain code of the ROI results in the initial snake around the object's contour. The attention module supervises the results of the automatic initialization and the tracking itself.

Now the object tracking module can start (section 4.4). To reduce the computation time we imitate the foveal vision system of humans and we only look on a part of the image which would correspond to a 256×256 subimage in the maximal resolution image (768×576) containing the moving object. This is accomplished by a continuous update of the frame grabbing device registers.

To track over long image sequences one has to detect whether tracking is no longer possible, and a signal has to be sent to the motion detection module. So we can start tracking another object or try to locate and track the same object again. In Sect. 4.5 we will discuss possible and typical errors during the tracking, and we will present features which can be extracted out of the active contour to detect theses errors. Figure 3 gives an overview of the motion detection and tracking modules and the communication paths between them. The attention module supervises the whole system and controls the active parts.

4.2 Motion Detection

For tracking moving objects one first has to detect motion in the scene. Then one can segment the motion field into moving objects. For motion detection many algorithms exist, for example, feature based methods, optical flow, or correlation algorithms. With the exception of the feature based approach these methods are hardly suitable for real time motion detection without specialized hardware. The simplest and fastest motion detection algorithm is the computation of the difference image between consecutive images of an image sequence. With this algorithm regions in the image are detected in which changes of the gray values occur. These changes may be based on the moving camera, moving objects or noise in the image. Now assuming a static camera and only one moving object the greatest changes are produced by the moving object itself. To reduce sensor noise we use a threshold operation to get a binary region. Gaps in this region are closed by applying a mean filter on the image. As a result one gets a binary image — the region of interest (ROI) — which contains a moving object. To handle small ROI's which result from noise ROI's with a size below a threshold are neglected. Finally we compute the chain code of the remaining ROI as a representation of this region which is needed for the automatic initialization of the snake (see Sect. 4.3). In the following, this algorithm is shortly summarized:

Given a sequence of images $f_0(x,y),\ldots,f_n(x,y)$ of image size 768×576 we proceed in the following way (the algorithm is started at $t = 0$):

1. Downsampling of the images $f_t(x,y)$ and $f_{t+1}(x,y)$ to an image size of 128×128.

2. Compute the difference image $D_{t+1}(x,y)$ where

$$D_{t+1}(x,y) = \begin{cases} 0 & ; \quad |f_t(x,y) - f_{t+1}(x,y)| < \delta \\ 1 & ; \quad \text{otherwise} \end{cases} \tag{10}$$

3. Close gaps and eliminate noise in the difference image using an appropriate filter operation (for example a mean filter, or a Gaussian filter; we use a 5×5 mean filter twice) to get the attention map $D^{att}_{t+1}(x,y)$. The set of interesting points in the image — the region of interest — contains the points (x,y) with $D^{att}_{t+1}(x,y) = 1$.

4. If there is no significant region, i.e. the area of the ROI is less than a given threshold, we assume that there is no moving object. Increment t by one and go to step 1.

5. Extract a chain code for the boundary of the binary region of interest.

6. If the features (for example the moments, or the area) of the extracted region differ from the previous region in a significant manner, then take the next image and go to step 1; (this means that the object is moving into the field of vision of the static camera).

The described step of motion detection can be seen in the upper left part of Figure 3. The result is a ROI marked as a black region. Additionally in step 6 of the algorithm a signal can be sent to the robot control to perform a saccade, i.e. a fast motion of the camera to another point in space to get the ROI in the middle of the image.

As mentioned earlier for the motion detection the camera remains static. In our future work we will extend this module to detect moving objects during movements of the camera as described for example in [20]. Thus, at present it is not necessary to realize the motion detection module in a separate process. We have implemented this module in one process together with the tracking and attention module due to the fact that at this time these modules may work sequentially. This is not a real limitation, because as soon as we extend the motion detection to work on images grabbed with a moving camera, we can easily realize this module as a process working in parallel to recognize other moving objects during the tracking.

4.3 Automatic Initialization of Snakes

As a result of the motion detection we get a region represented by a contour as a chain code. Assuming that there is a moving object inside the ROI we now have to initialize the snake so that it is close to the contour of the moving object.

This initialization has to be computationally efficient, too. In the literature one can find an initialization by the tangent field [26]. But this method is computationally expensive and therefore it cannot be applied to real time closed loop applications. Most of the other authors use an interactive initialization.

For initialization we make use of one property of non-rigid snakes. In the case of low or zero external energy — that means no edges are near the snake — the snake will collapse to one point. This can be easily seen in equations 2 and 3. The sum of the discrete derivatives is zero, and therefore minimal, if all positions $\boldsymbol{v}(s)$ of the snake elements are equal. Because of this fact it is sufficient to do a coarse initialization around the object's contour. The snake will collapse until it reaches the contour of the moving object. As a result of the motion detection module we get the chain code of the ROI. This chain code is used as the initial snake.

One of the possible errors this idea suffers from is the presence of strong background edges near the moving object. In this case the active contour will slither to such edges and will not extract the moving object itself. But we will show in our experiments that this initialization works well in our current system. If the attention module (see Sect. 4.5) detects an error in the initialization it sends a signal to the motion detection module. Otherwise, after the initialization the motion tracking module will start as it is described in the next section.

4.4 Object Tracking

The principle of tracking a moving contour, i.e. a moving object, is clarified in Figure 4: In an image f_t the snake converges to the contour of an object. In image f_{t+1} the snake is placed at the position reached in image f_t. In addition, a prediction of the movement of the object in the 2D image plane can be performed and so the snake shifted to this predicted position in f_{t+1}. Assuming that the distance of the object to the snake in pixels is sufficiently low the snake will again slither down into the minimum of the external energy computed out of the object's contour.

In Figure 5 the algorithm describing the object tracking is shown. There exist approaches using a prediction step for moving the snake to the predicted position of the object in the next image [7, 25]. Such a prediction step may be for example based on a Kalman filter. Thus the tracking can be made more robust, and one can allow faster motion of the object. At present, we use no prediction step in our work. In our approach we shift a small window (256×256) over the physical image to track the moving object. The advantage of using a small window with full resolution over using a low resolution full camera image is that we get more details of the object and more exact edges. Also, we can smooth the external energy by a larger filter. The smoothness of the external energy (i.e. the edges of the moving object) influences the maximum displacement of the object between successive images. In our environment this maximimum displacement is 8 pixels. With this approach we cannot track outside the field of view of the static camera, and we cannot keep objects in the middle of the image which move

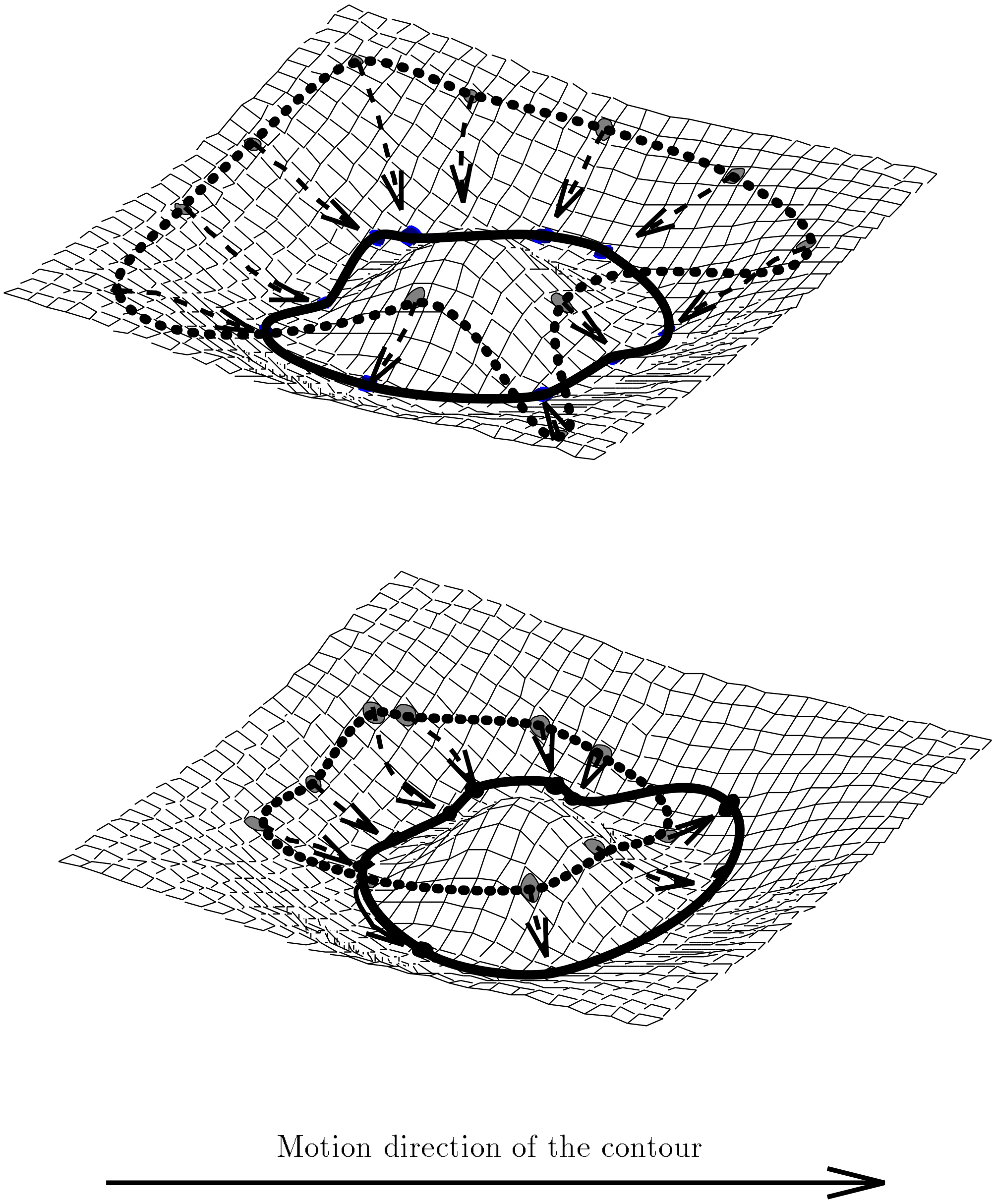

Figure 4: Tracking principle using active contour models (compare Figure 1). Top: The initial snake (dotted line) and the result of energy minimization (solid line). Bottom: Extraction of the contour in the next image. Result of the previous image (dotted line) converges to the contour of the circle (solid line), if the displacement of the object in the image plane is less than a maximum displacement given by the smoothing of the energy and the object's size.

at the border of the field of view. For that we use the robot's ability to move the camera (see subsection 4.6).

The parameters of the snakes are kept constant during all experiments. We choose $\alpha = 0.1$, $\beta = 1$ and $\gamma = 10$ heuristically (see equation 3, 8 and 9). The number of iterations was fixed to 200. Our experiments show that after 200 iterations the snake always converged to the object's contour. In almost all of the images the snake converges much faster. That means,

```
snap the start image $f_0(x,y)$ (full optical image size, resolution $128 \times 128$)
put the active contour $\boldsymbol{v}(s)$ around the moving object in image $f_0(x,y)$,
$f_{act}(x,y) = f_0(x,y)$
WHILE snake has not lost the object
    compute external energy $E_{ext}$ out of $f_{act}(x,y)$
        minimize energy $E(\boldsymbol{v}(s)) = E_{int}(\boldsymbol{v}(s)) + E_{ext}(\boldsymbol{v}(s))$
    UNTIL snake $\boldsymbol{v}(s)$ converges
    snap image $f_{act}(x,y)$ in a way that the center of gravity of the snake
    is in the middle of the subimage
    (window out of the camera image: $256 \times 256$ pixels)
```

Figure 5: Algorithm describing the tracking with active contour models.

that the computation time can be improved by extraction a convergence criterion. An idea for such a measure can be found in [18].

As a result of the tracking algorithm one gets the center of gravity of the snake which is the center of gravity of the moving object if the snake is assumed to cover the object's contour. The center of gravity is now used to change the grabber's parameters in a way that the center of gravity will be in the middle of the image. We use the center of gravity to determine the window out of the camera's optical image to be digitized. That means, no other image data of the scene is available, and if we have lost the moving object there is no way to find the object again. Thus, such errors (for example losing the object) have to be detected to reset the camera's parameters to get the full image data out of the A/D converter on the video card. This is done by the attention module described in the next subsection. Simultaneously, we send the center of gravity to the robot control to move the robot if the moving object comes close to the border of the physical camera image.

4.5 Feature Based Detection of Errors by the Attention Module

For tracking we only digitize a small window (256×256) out of the camera's image. Thus the single steps of our system can be computed very fast because of their simplicity and the reduced amount of data which must be processed. Due to this simplicity, errors may occur. For example, the ROI may not contain a moving object. So the collapsing snake does not extract a moving object but some edges or contours of the background. During tracking the object may stop moving, or it moves behind something in the background, or outside the field of view of the camera. In addition, the snake may lose the object's contour, if the displacement of the object in pixels between successive frames is too large. To get a robust tracking over long sequences of images we have to detect such errors. Then the attention module will start the motion detection module which will look at the full camera image to detect a moving object again.

We have investigated some features which can be used for the detection of such errors described above. If the active contour extracts a static background object one cannot measure any motion of the snake. As mentioned earlier, if the initialization fails and there is no background edge near the snake, the snake will collapse to one point. If the snake extracts a background edge, it may degenerate to a line. To detect such events the following features can be extracted out of the active contour:

- Correlation:
  By measuring the correlation of the snake elements the degeneration to line can be detected.

- Regression:
  This feature is based on the same idea as the correlation.

- Moments of the contour:
  Looking at the moment of the contour one can detect, if the active contour degenerates to one point in the absence of edges, if the snake becomes a line, and if part of the snake extract the object and another part get caught by a background edge (i.e. the snake will be stretched). For this reason, we use the moments to detect errors in our experiments.

In Figure 6 the plots of the x-moment and y-moment of an active contour are shown during a sequence of images. Over the whole sequence the aspect of the toy train varies, because of the motion on a circular rail (see Sect. 5). The reason for the rapid change of the x- and y-moments which can be seen is that three snake elements extracted the rail instead of the train itself for 10 images. In this case from one image to the other the forces of the other snake elements extracting the train become so strong that the three snake elements are pushed away from the rail back to the moving object. This can be seen in the derivatives of the moments. Preliminary experiments showed that the derivative of the moments is a better feature than the moments itself, because this feature is more independent to the object's size.

In our experiments we use an upper and a lower threshold for the moments. If one of the moments violates the upper or lower threshold — that means the snake degenerates — we switch back to the motion detection module. This occurs if more than three elements extract one background edge (the snake becomes a straight line) or the snake falls into one point. We have also tested the correlation and regression but the best results could be achieved by the moments. A combination of the features did not lead to an improvement, because a straight line detectable by the regression coefficient or the correlation can also be recognized by one of the moments. As mentioned at the end of Sect. 4.2 the attention module is realized in one process together with the tracking and object detection module. At this time the attention module only watches over the result of the tracking, i.e. it judges the shape of the active contour. This can be done directly after the tracking step. In the future, as soon as the attention module is used to perform a knowledge based verification of the tracking, which cannot be done in real–time, we will realize this module as a separate process. This is also necessary as soon as we drop the assumption that only one moving object is in the scene. Then the attention module has to decide which object should be tracked (for example: always the closest to the camera). Our system design described in Sect. 3 allows such an extension.

4.6 Robot Control

The fourth part of our closed loop object tracking system is the module for robot control. There exist many approaches in the literature for robot control [22]. Since in a closed loop

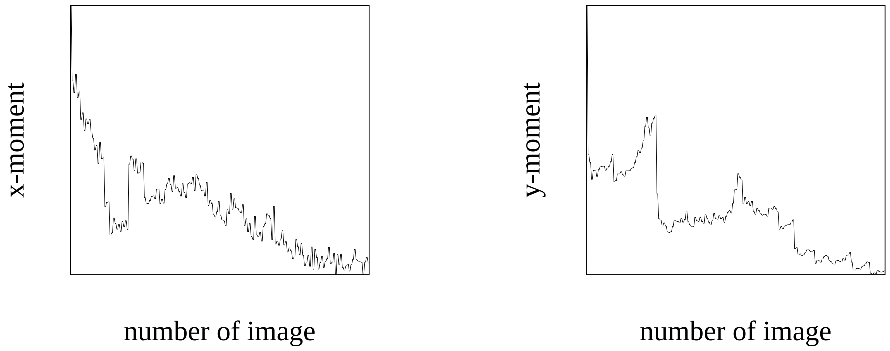

Figure 6: Features for error detection during tracking: the x- and y-moment of the active contour.

system the slowest module constrains the overall system performance the robot control should be kept as fast as possible. Therefore, we use only four signals for each direction in which the robot can move: + or − and `FASTER` or `SLOWER` (+ and − decide the direction of the movement of one of the axis). To cause a smooth robot motion we use two different methods for changing the grabbed image. First, we move the robot itself. Second, as mentioned in Sect. 4.4 we change the parameters of the grabbing device such that the tracked object is kept always in the middle of the subimage (see Figure 7). The tracking indicated by the dashed line is done by changing the area which the grabbing device will digitize. The motion of the whole image is indicated by the solid lines and is achieved by the robot's motion. In the tracking module the registers of the frame grabber card are set so that the snake, i.e. the moving object, is in the middle of the digitized image. Now, if the object moves near the border of the physical camera image, the window cannot be chosen such that the object is in the middle. Thus the tracking module sends the position of the center of the snake to the robot control. The control module uses a simple proportional controller [16] to move the robot. In the time domain the controller can be described by

$$\boldsymbol{y}(t) = V(\boldsymbol{x}_1(t) - \boldsymbol{x}_2(t)). \tag{11}$$

$\boldsymbol{y}$ is the control value, $\boldsymbol{x}_1$ the coordinate of the center of the physical camera image, and $\boldsymbol{x}_2$ the center of the active contour. This coordinate is sent to the robot control by the tracking module. To avoid oscillations of the robot around the position which should be reached we use a long delay time. This means that the moving object is not always in the middle of the camera image but always in the middle of the digitized window as mentioned earlier.

5 Experiments and Results

5.1 Experimental Environment

In this section we present our experimental environment and a qualitative measurement for judging the results of the experiments. Using the qualitative measurement we can show that

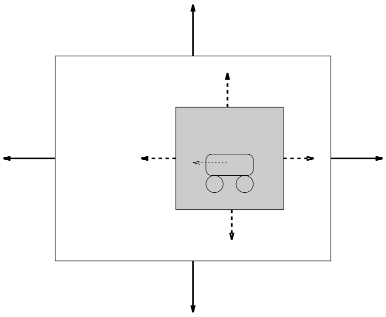

Figure 7: Two methods are used for changing the view of the scene: dashed lines indicate changing the parameters of the grabbing device; solid lines indicate the motion of the robot using a proportional controller. The large light rectangle is the field of view of the camera, the small gray rectangle is the area digitized on the board.

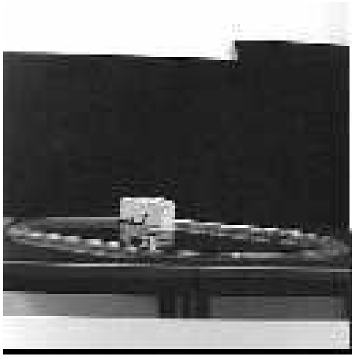

Figure 8: Toy train moving on a circle in front of the robot.

our tracking is robust and exact. Also we can show that the automatic initialization presented in Sect. 4.3 is sufficient in the context of our system.

In our experiments we have a moving toy train in front of a robot at a distance of about 1.5–2.0 meter (see Figure 8). Mounted on the robot's hand is a camera looking on the toy train. The toy train is moving on a circle with a speed of 2.0 cm/sec. Due to the hardware limitations (we get a maximum frame rate of 3 images per second) this speed corresponds to the maximum displacement (8 pixels) of the object in the image plane which we can track without prediction (see Figure 4). During the experiments we have constant lighting conditions and all parameters of the algorithms are kept constant. Other preliminary experiments showed that even drastic changes in the illumination cause no problem to the algorithm.

To judge automatically long sequences of images with various moving objects one has to use a measurement which can be computed out of the images after the experiments. We use the following measurement: Compute the center of gravity of the moving object (with a heuristic method suitable only for this experimental setup) and compare this coordinate with the center of the image. Because the camera window will be placed in a way that the center

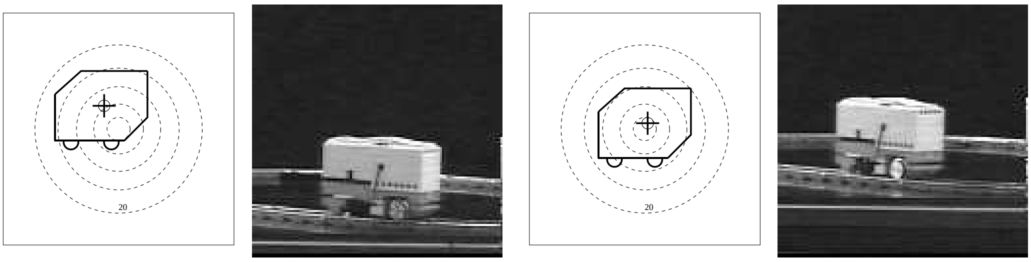

Figure 9: Qualitative judgement (from left to right): the principle of the measurement of a wrong and an exact tracking; two examples of images, one with a mediocre tracking result, the other one with an exact tracking.

of gravity of the snake is in the middle of the image, the center of gravity of the object also should be in the middle of the image, if the snake correctly covers the moving object. Now we measure the distance between the center of the image and the center of the object and use this distance as a qualitative judgement (see Figure 9).

As mentioned earlier, we realized the modules in two separate processes. The first is the robot control, the second one contains the motion detection, object tracking, and attention module. At present we use two workstations connected with a standard network (Ethernet). The robot control is distributed via the `pvm` on the workstation which is physically connected to the hardware of the robot. Most of the work is done on the second workstation on which the second process runs. The missing balance between the loads of the two workstation is no disadvantage of the system, because there arises no bottleneck between the two workstations.

5.2 Results

We present the results of nine experiments. In each of the experiments we track the moving object until 600 images are grabbed. This corresponds to an average length of the experiment of 7 minutes. In each of the nine experiments the train started on a different position on the rail circle. So in the sum of these nine experiments the train has moved over the whole circle, i.e. all possible changes of the contour of the train – based on the different viewing angles – and directions of motion could be recorded. In Table 1 the results of the experiments are shown.

In experiment 9 at the beginning of the experiment the moving object is not in the field of view of the camera. Thus, the system has to wait until a moving object enters the field of view. This results in some wrong initializations of the snakes due to sensor noise (large changes of gray values).

Experiments 6 and 7 start with the moving object at the border of the field of view: In experiment 6 the toy train moves outside the field of view. Here we have tested the ability of the robot control to get the object into the middle of the image. In experiment 7 the toy train moves into the field of view and the system has to wait until the object is completely in the image. The results show that the system can handle these cases. In experiment 6 the system initializes the snake 9 times; because of large control values (see equation 11) which implies a fast movement of the robot to get the object in the middle of the image the snake

experiment	# switches between MT & MD	# images	$\triangle s$ $\leq$ 5	$\triangle s$ $\leq$ 10	$\triangle s$ $\leq$ 20
exper. 1	0	600	20.0	50.0	95.5
exper. 2	0	600	38.2	81.8	100.0
exper. 3	2	600	2.5	8.2	61.4
exper. 4	5	600	9.4	35.8	91.1
exper. 5	2	600	12.7	32.2	86.4
exper. 6	9	600	14.2	44.7	93.9
exper. 7	2	600	28.6	59.6	90.3
exper. 8	2	600	16.9	36.0	89.8
exper. 9	0	600	22.5	54.9	74.5

Table 1: Results of the tracking: experiment, the number of switches between motion tracking (MT) module and motion detection (MD) module, the number of images tracked, and the amount of images in percent in which the distance of the center of gravity from the middle of the image is less than 5, 10, 20 pixels

loses the object. As soon as the moving object is no longer at the border of the image the initialization works well and the object can be tracked.

The worst experiment is experiment 3. In this experiment the attention module does not detect the error of the tracking. The reason is that we use two static thresholds for the detection of errors based on the moments of the contour described in Sect. 4.5. In this experiment these static thresholds do not allow us to detect the errors. Preliminary experiments showed that using the derivatives of the moments instead of two static thresholds makes the error detection more robust.

As described in Sect. 4.4 the subsequent image is digitized such that the center of gravity of the snake (which should be the center of gravity of the moving object) is in the center of the grabbed image. Because of the motion of the object between two consecutive images (4–8 pixels) the object cannot be perfectly in the middle of the image. So we define that an object is in the middle of the image when its center of gravity differs from the center of the image by less than 20 pixels. With this definition in five of the nine experiments – in over 90 percent of the images – the object is in the middle of the image.

The reason of the bad results during experiment 3 can be seen in Figure 10. The active contour has extracted the moving object but some contour elements extract parts of the rail, too. So, the center of gravity of the snake is in the middle of the image but the object is not. This result shows that the qualitative measurement is suitable to detect errors in the motion tracking module. In Figure 11 an example of a normal tracking result is shown. In Figure 12 parts of the sequence of images of experiment 2 can be seen. In all of the images the moving train is kept in the middle of the image. In column 2 of Table 1 the stability of the automatic initialization and tracking with the active contour can be seen. A switch to the motion detection module will be done in the case of an error in the tracking module or after the automatic initialization. Because of that, the initialization and tracking is robust as

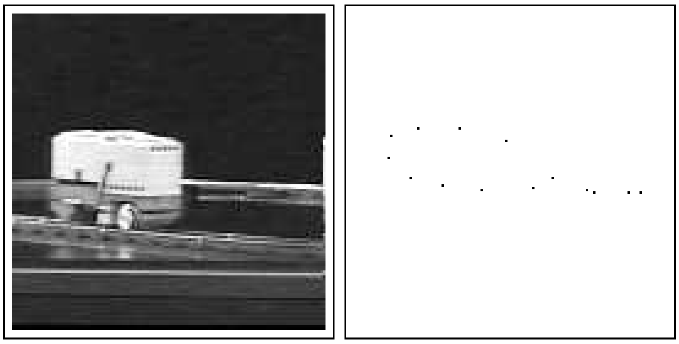

Figure 10: Worst case: experiment 3. Left the image, right the active contour. The active contour extracts parts of the rail, too.

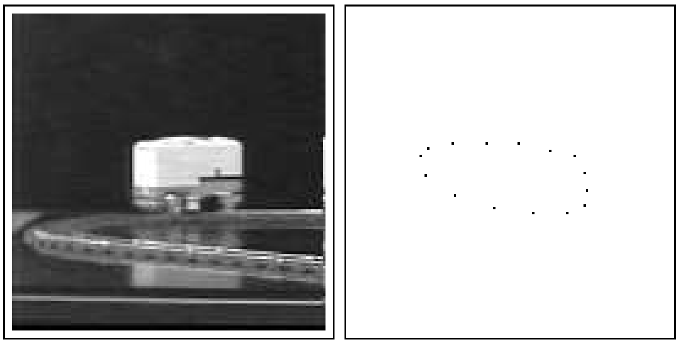

Figure 11: Correct result: experiment 1. Left the image, right the active contour.

one can see in the examples. In experiment 1, 2 and 9 the train is tracked after initialization without any error over 600 images.

Our experiments show that a lot of information about the motion of the train can be collected during the tracking. So the tracking itself could be made more robust by using a prediction step in the tracking module which is presently not realized.

In Figure 12 (third row) another experiment is shown. An image sequence of 100 images is processed with our system. Due to the limitations of our frame grabbing device (frame rate of 3 images/second) and the fact that a displacement of a maximum of 8 pixels can be achieved only with a high frame rate this sequence has been processed off-line. Therefore, also no movement of the robot has been possible but as one can see the active contour extracts the head sufficiently. Thus, it is obvious that also the camera can be steered to keep the head in the middle of the image. During the whole sequence the head is not lost but – as shown in image 91 – in 49 images the snake did not fully extract the head which corresponds to an error of 50 percent. The reason for the this result compared to the tracking of the train is the displacement between consecutive frames, the shadows near the cheek, and the mustache. Some of the errors are based on the image size which is in some frames too small to fully contain the head.

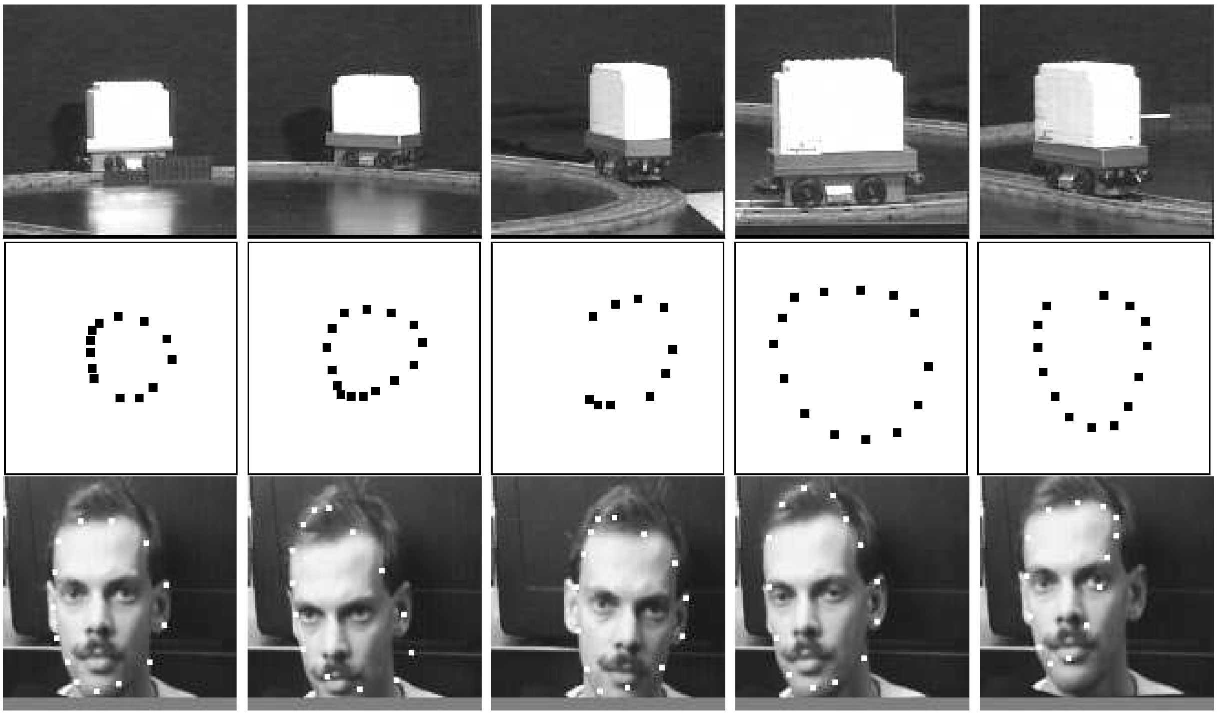

Figure 12: Images 13, 84, 207, 329, 424 of experiment 2. First row: The subimage containing the tracked object. Second row: the corresponding snakes. Third row: another experiment showing the suitability for telecommunication. Tracking of a head with active contours. Images 0, 45, 57, 69, 91 and the active contours are shown.

6 Discussion and Future Work

At this time the described approach still has some disadvantages: First, for the motion detection module a static camera is assumed. Second, the initialization of the snake is sensitive to noise in the image, for example to shadows of the moving object. Third, no prediction of the motion of the object is used during the tracking. For this reason, the tracking fails if occlusions occur or if strong background edges appear near the moving object. Thus, the class of scenes we have used contain one moving object in front of a homogeneous background. Of course, it causes no problem to allow more than one moving object. In this case the attention module has to decide which object should be tracked. This is not realized in our system at this time. Additionally, more than one snake can be put on one image to track more than one object simultaneously.

Additional experiments were carried out but not presented in this paper; they showed that the approach can be used for many different objects without changing the parameters of the algorithms. One example is given by the head sequence in Figure 12. The system is also robust with respect to the lighting conditions – even during the tracking – and the shape of the object (smooth like the head, and polyhedral like the train).

Other authors show that the tracking quality can be improved if a prediction is used [7, 25]. We verified this by preliminary experiments. We are working on a prediction step and we expect to get results in the near future. Up to now, partially occlusions of the moving object result in a loss of the contour. Thus, the experimental environment is limited to scenes without strong background edges.

While tracking the object's contour one can collect a lot of information about the shape of the object and the object itself (for example the color, or some significant features like corners). This information could be used to constrain the nonrigid snake used in our approach to a rigid snake model during the tracking. Additionally one can look for characteristic features of the moving object, for example to work on a special color channel if that channel contains significant information about the object.

Also we will do work on optimizing the speed of the algorithms and adjust some of the algorithms for preprocessing to the active vision domain. A lot of computation time can be saved if we can detect whether the snake has converged to the object's contour. An idea for such termination is given in [18].

In our future experiments we will verify the quality of the system in the case of natural background.

7 Conclusion

For real time motion detection and tracking in a closed loop of vision and action one has to reduce the data to be worked on. The new paradigm of active vision gives a lot of ideas and principles that can be used in real time image processing in general. In the field of motion tracking one class of tracking methods is the so called active contour model. Some authors have presented promising results in object tracking using snakes but there exists no system using snakes for real time object tracking without any specialized hardware like transputer networks or preprocessing hardware cards and without any hardware dependent software. In our work we use only standard *UNIX*© workstations and we implement all algorithms in an object oriented programming language (hardware independent).

We have presented a combination of several simple and fast working vision modules for object tracking. In the first module the motion is detected in the scene assuming a static camera. In the second module the moving object will be tracked using snakes. A third module is responsible for the control of the camera, i.e. steering a robot's arm with a mounted on camera, and determining a window in the camera image which will be digitized to keep the object in the middle of the image. All modules may work in parallel. For implementational reasons the motion detection and tracking module run in one process. Up to now most of the authors use an interactive initialization of the snake on the first image of a sequence of images. For real time closed loop object tracking such an initialization is not useful. Therefore, we have proposed an automatic initialization which is based on the difference image. The experiments show that this initialization is fast and robust. To improve the robustness of the overall system we extract features out of the active contour. This is done in an additional module the so called attention module. With these features we can detect errors in the initialization and the tracking itself. In the case of an error the attention module stops the camera motion and switches back to the motion detection module. In 90 percent of all images the moving object is in the middle of the image if one defines that the object is in the middle of the image if it is less than 20 pixels from the center of the image. For such judgement we have defined a qualitative measurement based on the distance of the center of gravity of the object and the middle of the image. We have presented real time experiments in a closed loop of vision and action which show that by combination of a tracking module and a motion detection module moving objects can be robustly tracked over long sequences of images in real time. For this purpose we do not use any specialized hardware and we do all implementations in an object oriented programming language leading to a very easily portable system.

References

[1] J. Aloimonos and D. Shulman. *Integration of Visual Modules.* Academic Press, Boston, San Diego, New York, 1989.

[2] J. Aloimonos, I. Weiss, and A. Bandyopadhyay. Active vision. *International Journal of Computer Vision*, 2(3):333–356, 1988.

[3] Y. Aloimonos. What I have learned. *Computer Vision, Graphics, and Image Processing*, 60(1):74–85, 1994.

[4] A. Amini, T. Weymouth, and R. Jain. Using dynamic programming for solving variational problems in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(9):855–867, 1990.

[5] R. Bajcsy and M. Campos. Active and exploratory perception. *Computer Vision, Graphics, and Image Processing*, 56(1):31–40, 1992.

[6] A. Benson and D.J. Evans. A normalized algorithm fir the solution of positive definite symmetric quindiagonal systems of linear equations. *ACM Transactions on Mathematical Software*, 3(1):96–103, 1977.

[7] M. Berger. Tracking rigid and non polyhedral objects in an image sequence. In *Scandinavian Conference on Image Analysis*, pages 945–952, Tromso (Norway), 1993.

[8] A. Blake and A. Yuille, editors. *Active Vision.* MIT Press, Cambridge, Massachusetts, London, England, 1992.

[9] L.D. Cohen and I. Cohen. Finite-element method for active contour models and balloons for 2–d and 3–d images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1131–1147, 1993.

[10] J. Denzler, R. Beß J. Hornegger, H. Niemann, and D. Paulus. Learning, tracking and recognition of 3D objects. In V. Graefe, editor, *International Conference on Intelligent Robots and Systems – Advanced Robotic Systems and Real World*, volume 1, pages 89–96, 1994.

[11] V. Fischer. *Parallelverarbeitung in einem semantischen Netzwerksystem für die wissensbasierte Musteranalyse.* PhD thesis, Technische Fakultät der Universität Erlangen-Nürnberg, (in preparation) 1995.

[12] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam. PVM 3.0 user's guide and reference manual. Technical Report ORNL/TM-12187, Engineering Physics and Mathemaics Division, Oak Ridge National Laboratory, Tennessee, 1993.

[13] W.A. Halang, G. Hommel, and R. Lauber. Perspektiven der Informatik in der Echtzeitverarbeitung. *Informatik–Spektrum*, pages 357–362, 1993.

[14] P. Karaolani, G.D. Sullivan, and K.D. Baker. Active contours using finite elements to control local scale. In D. Hogg and R. Boyle, editors, *British Machine Vision Conference 1992*, pages 481–487, London, Berlin, 1992. Springer.

[15] M. Kass, A. Wittkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 2(3):321–331, 1988.

[16] W. Leonhard. *Einführung in die Regelungstechnik.* Vieweg, Braunschweig, Wiesbaden, 1992.

[17] S.P. Levitan, C.C. Weems, A.R. Hanson, and E.M. Riseman. The UMass image understanding architecture. In L. Uhr, editor, *Parallel Computer Vision*, pages 215–248. Academic Press, Boston, Orlando, 1987.

[18] F. Leymarie and M.D. Levine. Tracking deformable objects in the plane using an active contour model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(6):617–634, 1993.

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

[20] D. Murray and A. Basu. Motion tracking with an active camera. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5):449–459, 1994.

[21] H. Niemann. *Pattern Analysis and Understanding*, volume 4 of *Series in Information Sciences.* Springer, Berlin Heidelberg, 1990.

[22] N.P. Papanikolopoulos, P.K. Khosla, and T. Kanade. Visual tracking of a moving target by a camera mounted on a robot: A combination of control and vision. *IEEE Transactions on Robotics and Automation*, 9(1):14–34, 1993.

[23] D.W.R. Paulus. *Objektorientierte und wissensbasierte Bildverarbeitung.* Vieweg, Braunschweig, 1992.

[24] M.J. Swain and M. Stricker. Promising directions in active vision. Technical Report CS 91-27, University of Chicago, 1991.

[25] D. Terzopoulos and R. Szeliski. Tracking with Kalman snakes. In A. Blake and A. Yuille, editors, *Active Vision*, pages 3–20. MIT Press, Cambridge, Massachusetts, London, England, 1992.

[26] S.W. Zucker, C. David, A. Dobbins, and L. Iverson. The organization of curve detection: Coarse tangent fields and fine spline coverings. In *Second International Conference on Computer Vision, Tampa, Florida*, pages 568–577, 1988.