A Two Stage Real–Time Object Tracking System

J. Denzler, H. Niemann
Lehrstuhl für Mustererkennung (Informatik 5)
Universität Erlangen–Nürnberg
Martensstr. 3, D–91058 Erlangen, Germany
email: {denzler,niemann}@informatik.uni-erlangen.de

denzler@informatik.uni-erlangen.de

The following paper will appear in the
Proceedings on the 2nd German–Slovenian Workshop
on
Image Analysis

Ljubljana, June 7th, 1994

A Two Stage Real–Time Object Tracking System

J. Denzler, H. Niemann
Lehrstuhl für Mustererkennung (Informatik 5)
Universität Erlangen–Nürnberg
Martensstr. 3, D–91058 Erlangen, Germany
email: {denzler,niemann}@informatik.uni-erlangen.de

Abstract

Active contour models (snakes) can be used for contour description and extraction, as well as for object tracking in image sequences. Two unsolved problems for real time object tracking are the problem of an automatic initialization of the snake on the object and the proof of robustness of this object tracking method.

In this paper we describe a two stage real time object tracking system. In the first stage, the moving object is detected and the active contour is initialized. In the second stage, the object is tracked by active contour models. The parameters of the camera and the frame grabbing device are continuously updated in such a way, that the moving object will always be kept in the center of the image. We show how features can be extracted out of the snake tracking the object which are used for detection of errors in the tracking stage. In this case the system switches back to the first stage for object localization.

We illustrate through examples that a robust tracking over long image sequences in real time is possible within this two stage system. For experimental evaluation of the tracking result, we present a formal error measure. Using this measure, the moving object is correctly in the center of the image in up to 95 percent of all images.

Keywords: active contour models, tracking, real–time, active vision

1 Introduction

In computer vision the field of real time image processing becomes more and more important today. Because of the increasing computation performance – a rule of thumb says, that every two years the computational power doubles – real time image processing can be applied to real world applications. For example autonomous vehicles, service robots, or machines assisting handicapped persons will be constructed in the next ten years. One important part for such machines will be 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.

For real time motion tracking it is necessary to have a closed loop of sensoric – which produces the image data – and action – for example moving the eyes to track a moving object. Moving the eyes implies that new image data will be brought into the system and therefore new action has to be performed. The calculation for this task has to be done in real time, because if a moving system detects an obstacle, actions to avoid this obstacle have to be taken before the system runs into the obstacle. In the field of object tracking the system has to compute the necessary motion of its eyes in a way, that after moving the eyes the object being tracked is again in the field of view.

In the last ten years a new paradigm called *active vision* came up in the field of computer vision [1, 2, 4, 10]. The main principle of active vision can be described by a definition of Aloimonos[1]:

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

In difference to that, the approach of Marr [8] says:

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

Some of the main technical mechanism of active vision are pyramids, attention modules, or in general selection in space, time and resolution [10].

In the last five years many authors have proven that using this new paradigm real time constraints can be satisfied, and promising algorithms and methods were constructed. One of these methods – especially for object tracking – is the so called *active contour model*, often also called *snake* [6]. In our problem domain, a toy train has to be tracked, moving in front of a robot. The robot has a camera mounted on its hand. The tracking should be performed is such a way, that the toy train is always in the middle of the camera image. As an additional constraint no specialized hardware for preprocessing, filtering

[1]stated on a talk about active vision given at KIFS'94

or segmentation will be used. All algorithms will be implemented on standard Unix workstations in an object–oriented programming language and environment [5, 9].

In the next section we will introduce the principles of active contour models and we will motivate the use of active contours for object tracking. In section 3 we will describe the ideas of our two stage real time object tracking system. After this, we will present our experimental environment. Real time 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 of the results and an outlook on further work that will be done in this context.

2 Active Contour Models

The energy minimizing model of active contours (*snakes*) was first introduced by [6]. 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 $\boldsymbol{v}(s)$

$$\boldsymbol{v}(s) = (x(s), y(s)),\ s \in 0, \cdots, n-1 \tag{1}$$

with

$$x(s) \in [0, x_{max}], \quad y(s) \in [0, y_{max}] \tag{2}$$

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

$$E \ = \ \sum_{i=0}^{n-1} \left(E_{int}(\boldsymbol{v}(i)) + E_{ext}(\boldsymbol{v}(i))\right). \tag{3}$$

E_{int} is mostly defined as (see [6])

$$E_{int}(\boldsymbol{v}(i)) = \frac{\alpha(i)|\boldsymbol{v}_s(s)|^2 + \beta(i)|\boldsymbol{v}_{ss}(s)|^2}{2} \tag{4}$$

$\boldsymbol{v}_s(s)$ and $\boldsymbol{v}_{ss}(s)$ are the partial derivatives of $\boldsymbol{v}(s)$. An interpretation of these terms can be given in the following way: Connecting two straight rails with a circle minimizes $\boldsymbol{v}_s(s)$ on the circle, but at the connections of the circle with the two lines a train would be derailed (see Figure 1 left). Smoothly connecting the two straight rails at the connections minimizes $\boldsymbol{v}_{ss}(s)$ but the train cannot move on the small radius (see Figure 1 right).

The external forces can be the image $f(x, y)$ or the edge strength of an image, for example $E_{ext}(\boldsymbol{v}(i)) = -|\nabla f(\boldsymbol{v}(i))|^2$. In the case of the edge strength as the external force

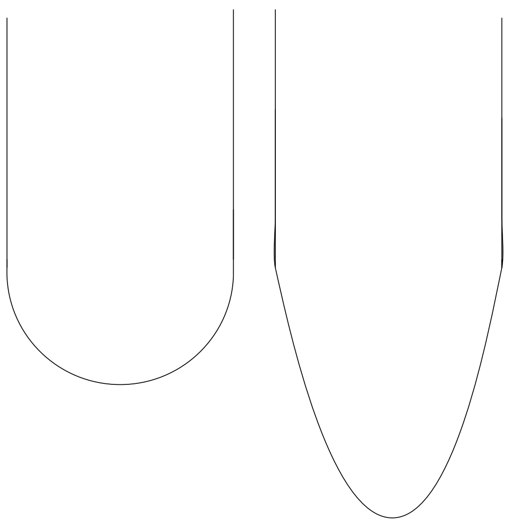

Figure 1: Left: A connection of two straight lines minimizing the first derivative of v(s). Right: A connection of two straight lines minimizing the second derivative of v(s) (like a hanging rope).

during the energy minimization the snake will be pushed to strong edges, for example the contour of an object. The principle of the active contour models is clarified in Figure 2; one can see a snake with seven snake elements positioned on an energy field. For example, this energy field could be computed of the negative edge strength of a circle using a standard edge operator, e.g. a Sobel operator. Now, during the energy minimaziation step each snake element slithers downhill to the next minimum. On this way, the contour of the circle will be extracted. Further information concerning the snake model and its behavior can be found in [6] and [7].

In several papers, for example [3, 7], the advantages of snakes for object tracking were shown. Given an image sequence $f_0(x,y), f_1(x,y), \ldots, f_n(x,y)$ including one moving 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 $f_{t+1}(x,y)$. 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.

This approach has several advantages over other feature matching algorithms for real time object tracking:

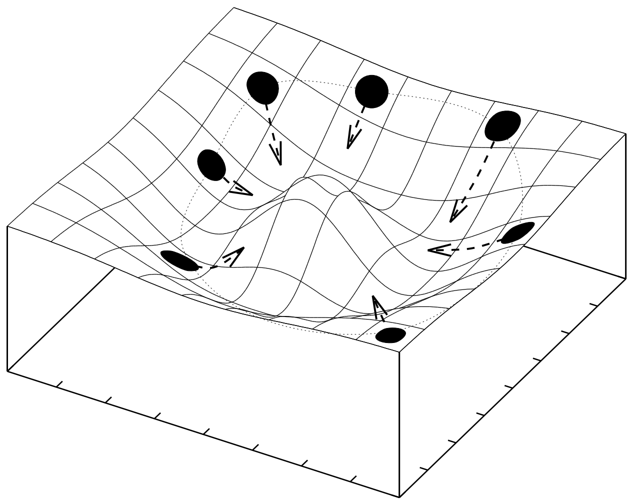

Figure 2: Principle of active contours: A snake with 7 elements extracts the contour of a circle, by moving into the minimum of the negative edge strength of the circle.

- Only small parts of an image have to be processed, namely a small region around the snake elements. In our approach the processing consists of edge extraction.
- Object extraction and object tracking is done in one step.
- There is no use of a foreground/background distinction. This is advantageous if the camera is moving, too.
- If the moving object is occluded, the snake is able to retain the object's contour for a while.
- 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 several parameters have to be chosen –in most of the cases heuristical. Second, the snake must be positioned close to the contour which should be tracked. This is mostly done in an interactive way. In the next section we present a method for an automatic placement of the snake near a moving object in an image sequence, grabbed with a static camera.

3 Two Stage Closed Loop Object Tracking

3.1 General Idea

The goal is to keep a moving object in the middle of the camera image. As a first approach we decompose the problem into two parts. First the motion in the scene has to be detected and one has to localize a region of interest, in which motion occurs (section 3.2). For that, the full optical image max size of 768×576 at a low resolution of 128×128 will be used

Then we have to initialize the snake on the object's contour. Up to now, all authors use an interactive initialization of the snake. Under real time constraints no automatic initialization exists. We will present a fast and in this context robust method for an automatic initialization. This will be described in section 3.3.

Now the object tracking stage can start (section 3.4). To imitate the foveal vision system of the humans and of course to reduce the computation time, now we only look on a part of the image which would correspond to a 256×256 subimage in the max 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 in stage 2. This may happen if the object stops moving, or the object moves behind something in the background, or the object moves outside the field of view of the camera. In addition the snake may loose the object's contour. In such cases we have to detect this event and then we will switch back to stage 1 for motion detection. So we can start tracking another object or locate and track the same object again. We will show that features can be extracted out of the active contour which can be used to detect errors described above (section 3.5). Figure 3 gives an overview of the complete system.

3.2 Stage 1: Motion Detection

For tracking moving objects one has first to detect motion in the scene. Then one can go on segmenting the motion into moving objects. For motion detection many algorithm exist. For example feature based methods, optical flow, or correlation algorithms. 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

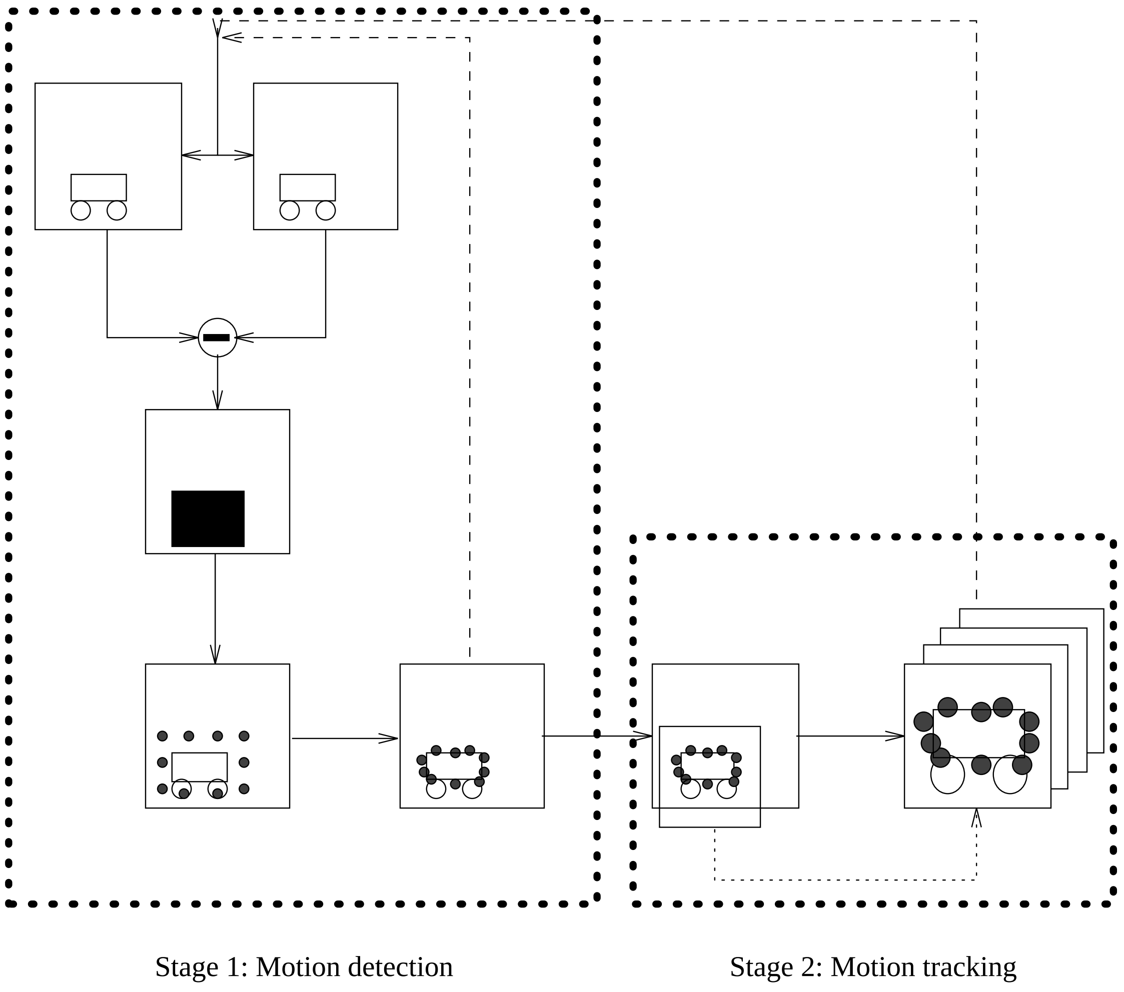

Figure 3: Overview of the 2–stage system.

object itself. To reduce sensor noise we use a threshold operation to get a binary region. Gaps in this region will be 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 consider small ROI's which result from noise, ROI's with a size below a threshold will be neglected. Finally we compute the chain code of the remaining ROI as a representation of this region, needed for the automatic initialization of the snake (see section 3.3). In the following, this algorithm will be 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 (start the algorithm at $t = 0$):

1. Down-sampling 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 $D_{t+1}(x,y) = f_t(x,y) - f_{t+1}(x,y)$, where

$$D_{t+1}(x,y) = \begin{cases} 0 & ; \quad |f_t(x,y) - f_{t+1}(x,y)| < \delta \\ 1 & ; \quad \text{otherwise} \end{cases} \tag{5}$$

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) to get the attention map $D_{t+1}^{att}(x,y)$. The set of interesting points in the image – the region of interest – contains the points (x,y) with $D_{t+1}^{att}(x,y) = 1$.

4. If there is no significant region, we assume that there is no moving object t. 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; (that means 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 rectangle.

3.3 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 have now to initialize the snake on the contour of the moving object.

This initialization has to be computationally efficient, too. In the literature one can find some automatic initialization by the tangent field [12]. But this method is very complicated and cannot be used in a real time closed loop application.

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 3 and 4. 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 stage we get the chain code of the ROI. This chain code will be used as the initial snake.

One of the possible errors this ideas suffers from, is the presence of strong background edges near the moving object. Then the active contour will slither to such edges and will not extract the moving object itself. But we will show in out experiments, that this initialization works well in our current system. In the case of an error in the initialization, the system switches back to the motion detection stage. This can be seen in Figure 3, in which a path exists back to the difference image from the end of the initialization phase. If the snake extracts the moving object after the initialization, then the system will switch to the motion tracking stage, described in the next section.

3.4 Stage 2: The 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 will be placed at the position reached in image f_t. Assuming that displacement of the object 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 is shown describing the object tracking. There exist some approaches, using a prediction step for placing the snake on the next image [3, 11]. Using such prediction step, for example a Kalman filter, 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. Up to now we do not steer the robot's arm to keep the object in the center of the camera image. In our approach we use a small window (256) out of the camera data, to track the moving object. The advantage 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 the object can do between successive images. A disadvantage of that approach is, that we cannot track outside the field of view of the static camera, and that we cannot keep objects in the middle of the image, which move at the border of the field of view.

The parameters of the snakes are kept constant during all experiments repeated here. We choose $\alpha = 0.1$, $\beta = 1$ and $\gamma = 10$. The number of iterations is fixed to 200. Our experiments show that after 200 iterations the snake has always converged to the object's contour. In most of all images, the snake converges much faster. That means, that the computation time can be improved by extraction a measure for the detection if the snake has converged.

As a result of the tracking algorithm one gets the center of gravity of the snake, which will be the center of gravity of the moving object, in the case that the snake covers the object contour. The center of gravity will now be used to change the camera parameters

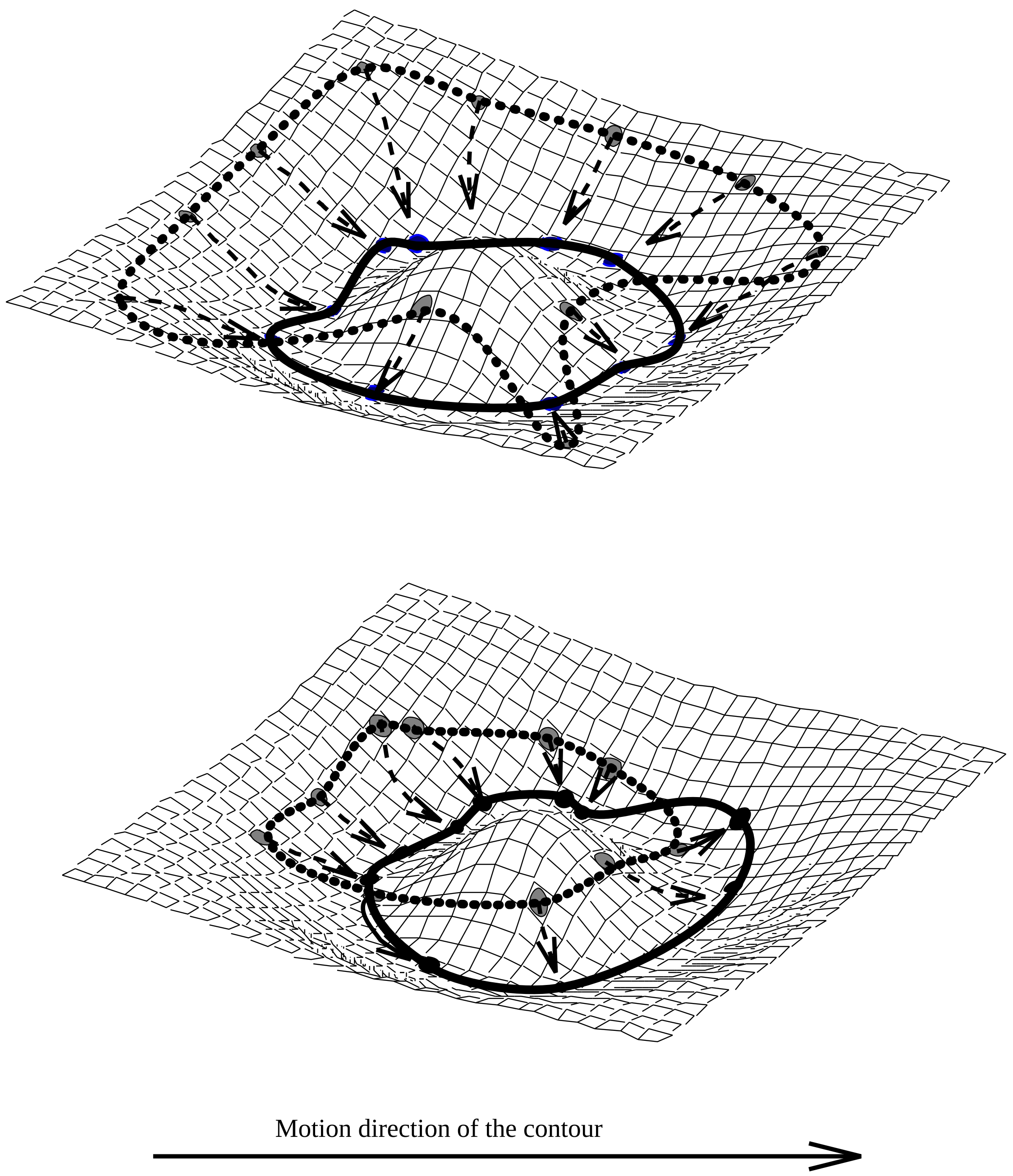

Figure 4: Tracking principle using active contour models.

in a way that the center of gravity will be in the middle of the image. At present we do not move the robot's arm. We use the center of gravity to determine a window out of the camera's optical image which will be digitized. That means, now other image data of the scene is available in our machine, and if we have lost the moving object, there is no way

```
snap the start image $f_0(x,y)$ (full optical image size, resolution $128 \times 128$)
put the active contour $\boldsymbol{v}(s)$ near 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 camera image: $256 \times 256$)
```

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

to find the object again. Instead we then have to reset the camera's parameters, to get the full image data out of the A/D converter on the video card.

3.5 Feature Based Detection of Errors During the Tracking Stage

In our approach we use a small window (256×256) out of the camera's optical image. The single steps of our system can be computed very fast because of their simplicity. Because of that, errors may occur. For example the ROI does not contain a moving objects. So the collapsing snake does not extract a moving object, but some edges or contours of the background. It is also possible that the snake loses the object during the tracking. To get a robust tracking over long sequences of images we have to detect such errors. Then we will switch back to stage 1, look at the full camera image to detect a moving object again.

In our investigations 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, then one cannot measure any motion of the snake. On the other hand if the initialization fails and there is no background edge near the snake, then as mentioned earlier the snake will collapse to one point. To detect such events the following features should be extracted out of the active contour: correlation, regression, moments of the contour.

In Figure 6 the plots of the x-moment and y-moment of the active contour are shown

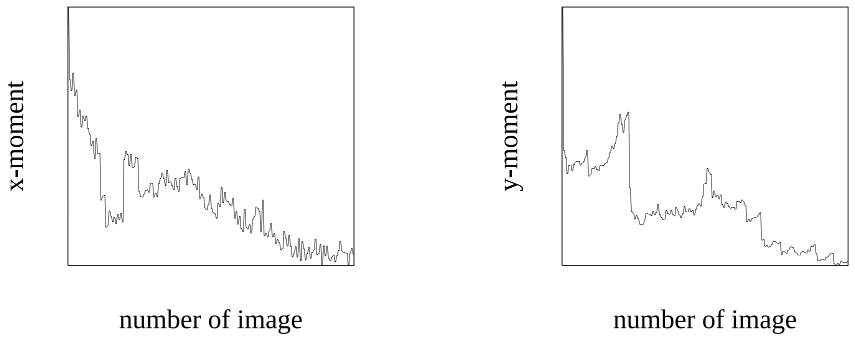

Figure 6: Features for detection of errors during tracking: the x- and y-moment of the active contour.

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 section 4). The reason for the rapid change of the x- and y-moments, which can be seen, is that three snake elements extract for 10 images the rail instead of the train itself. Then from one image to the other the forces of the other snake elements extracting the train becomes so strong that the three snake elements were pushed away from the rail back to the moving object. This can be seen in the derivative of the moments.

In out experiments we use an upper and a lower threshold. If one of the moments increases the lower or decreases the upper threshold – that means the snake degenerates – we switch back to stage one. This occurs if more than three elements extract the rail (the snake becomes a straight line) or the snake falls into one point.

Using theses features we have the ability to make such coarse initialization, because in the case of a wrong initialization one can detect the failure and can switch back and try the initialization again. In the next section out experiments will show, that first the initialization is robust and second if an error occurs, such error will be detected.

4 Experiments and Results

4.1 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 proof, that our tracking is robust and exact. Also we can show that the automatic initialization

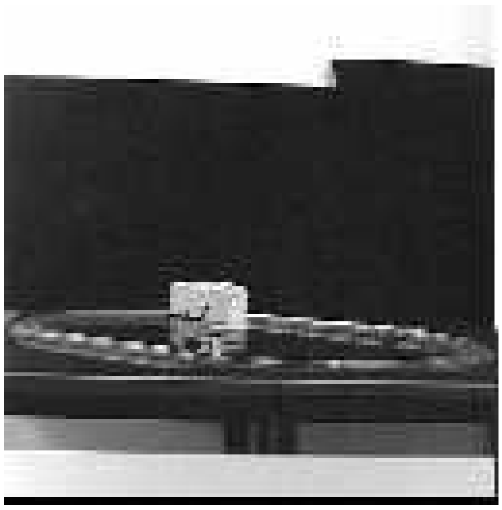

Figure 7: Toy train moving on a circle in front of the robot.

presented in section 3.3 is sufficient in the context of our system.

In our experiments we have a moving toy train in front of a robot (see Figure 7). Mounted on the robot's hand there is a camera, looking on the toy train. The toy train is moving on a circle with a speed of 1.2 cm/sec. During the experiments we have constant lighting conditions and all parameters of the algorithms were constant. Other experiments showed that even drastic changes in the illumination causes 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 and compare this coordinate with the center of the image. Because the camera window will be placed in a way that the center of gravity of the snake is in the middle of the image, the center of gravity of the object has to be too, if the snake correctly covers the moving object. Now we measure the distance between the center of the image (the point $(64, 64)$) and the center of the object and use this distance as a qualitative judgement (see Figure 8).

4.2 Results

We present here the results of five experiments. In experiment 1 to 4 we have tracked the object during 200 images. In each of the four experiments, the train started on a different position on the rail circle. So in the sum of these four experiments the train has moved

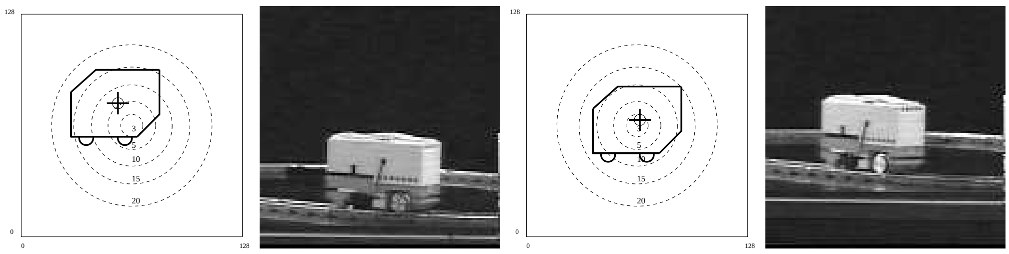

Figure 8: Qualitative judgement (from left to right): the principle of the measurement for a wrong and an exact tracking; two examples of images, one with a mediocre tracking result, the other with an exact tracking.

experiment	# switches between stage 2 & 1	# images	$\triangle s \leq 3$	$\triangle s \leq 5$	$\triangle s \leq 10$	$\triangle s \leq 15$	$\triangle s \leq 20$
exper. 1	4	195	21.5	80.5	96.4	97.9	99.0
exper. 2	1	172	16.3	40.7	50.0	63.4	79.0
exper. 3	2	126	27.8	93.6	98.4	99.2	100
exper. 4	7	189	1.0	9.5	61.9	85.2	92.6
exper. 5	0	354	13.8	35.3	60.5	98.6	100

Table 1: Results of the tracking: the experiment, the number of switches between stage 2 and 1, 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 3, 5, 10, 15, 20 pixels

over the whole circle, i.e. all possible changes of the contour of the train and directions of motion could be recorded. In experiment 5 the train has been tracked over 500 images, which corresponds to one complete motion on the whole rail circle. In Table 1 the results of the experiments are shown. If we assume, that an object is in the middle of the image if its center of gravity differs from the center of the image by less than 10 pixels, then in two of the five experiments – in over 95 percent of the experiments – the object is in the middle of the image. In the other experiments, the tracking result is worse. But with the exception of experiment 2, in over 90 percent of all images the moving object has a distance to the middle of the image by less then 20 pixels. The reason of the bad results

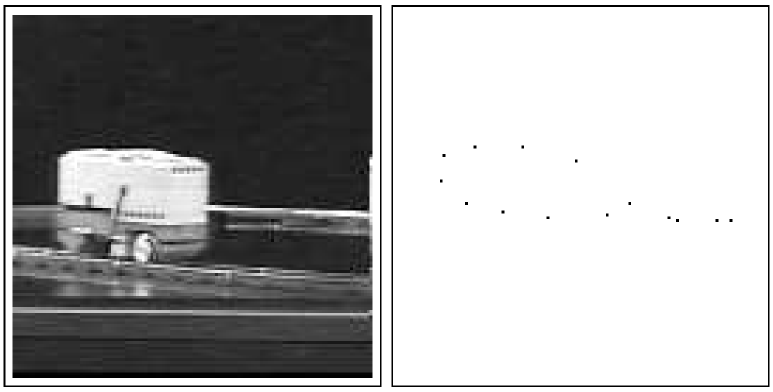

Figure 9: Worst case: experiment 2. Left the image, right the active contour. The active contour extracts parts of the rail too.

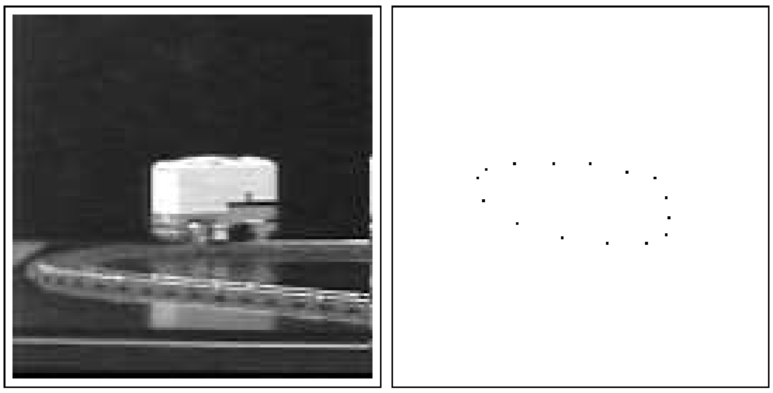

Figure 10: Normal result: experiment 1. Left the image, right the active contour.

during experiment 2 can be seen in Figure 9. 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 objects is not. This result shows, that the qualitative measurement is suitable to detect errors in the tracking stage. In Figure 10 one example of a normal tracking result is shown. In Figure 11 a long sequence of images of experiment 5 can be seen. In all of the images the moving train is kept in the middle of the image. In column 3 of Figure 1 the stability of the automatic initialization and tracking with the active contour can be seen. Because a switch to stage 1 will be done in the case of an error in the tracking stage or after the automatic initialization, one can see that the tracking is robust. In experiment 5 the train is tracked after initialization

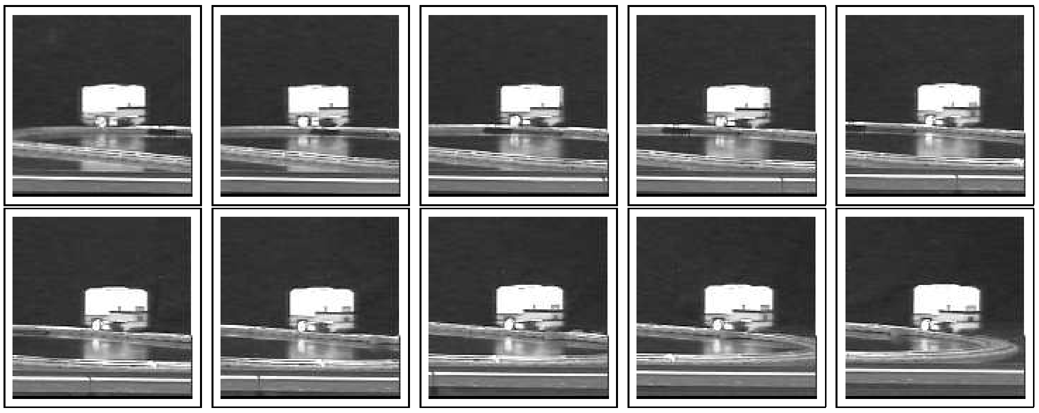
Figure 11: Result of experiment 5. Images 207, 227, 247, 267, 287, 307, 327, 347, 367 and 387 are shown.

without any error over 500 images.

It is again worth mentioning that at present no prediction step in the tracking stage is used. 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 stage.

5 Future Research

At this time the described approach still has some disadvantages. First for the motion detection stage a static camera is assumed. The initialization of the snake is sensitive to noise in the image, for example shadows of the moving object. Another disadvantage is that no prediction of the motion of the object is used during the tracking. For this reason, the tracking fails if occlusions occurs or if strong background edges appear near the moving object.

Other authors show that the tracking quality can be improved if a prediction is used [3, 11]. We are working on a prediction step and we will get results in the near future. Our preliminary experiments show that with a prediction step the system can be made more robust.

Another work to be done in the next future will be the steering of the robot. Then we perform a tracking by slow motions of the camera window and saccades (fast motions of the robot) if the object moves near the border of the camera's field of view.

Over the tracking of 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 can be used to constrain the nonrigid snake used in our approach to a rigid snake model during the tracking. Additionally one can look for such features, 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 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 [7].

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

6 Conclusion

For real time motion detection and tracking in a closed loop of sensoric 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 method 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 and preprocessing hardware cards. In our work we like to use only standard Unix workstations and we implement all algorithms in an object oriented programming language.

We have presented a two stage object tracking system using active contour models. In the first stage the motion is detected in the scene assuming a static camera. In the second stage the moving object will be tracked using snakes. For real time closed loop object tracking an interactive initialization of the snakes on the contour of the object in the first image as used by most of the authors up to this time is not useful. Therefore, we have proposed an automatic initialization. It 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. With the features, we can detect errors in the initialization and the tracking itself. In the case of an error, the system stops the camera motion and switch back to stage 1 for motion detection. Over all experiments, in 70 percent of all images the moving object is in the middle 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 shows, that in the combination of a tracking stage and a motion detection stage moving objects can be robustly tracked over long sequences of images in real time. For that we do not use any specialize hardware and we do all implementations in an object oriented programming language, for portability reasons.

References

1. J. Aloimonos, I. Weiss, and A. Bandopadhay. Active vision. *International Journal of Computer Vision*, 2(3):333–356, 1988.
2. R. Bajcsy and M. Campos. Active and exploratory perception. *Computer Vision, Graphics, and Image Processing*, 56(1):31–40, 1992.
3. M. Berger. Tracking rigid and non polyedral objects in an image sequence. In *Scandinavian Conference on Image Analysis*, pages 945–952, Tromso (Norway), 1993.
4. A. Blake and A. Yuille, editors. *Active Vision.* MIT Press, 1992.
5. 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 –*, to appear September 1994.
6. M. Kass, A. Wittkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 2(3):321–331, 1988.
7. 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.
8. David Marr. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information.* W.H. Freemantle, San Francisco, 1982.
9. D.W.R. Paulus. *Objektorientierte und wissensbasierte Bildverarbeitung.* Vieweg, Braunschweig, 1992.
10. M.J. Swain and M. Stricker. Promising directions in active vision. Technical Report CS 91-27, University of Chicago, 1991.
11. D. Terzopoulos and R. Szeliski. Tracking with kalman snakes. In A. Blake and A. Yuille, editors, *Active Vision*, pages 3–20. MIT Press, 1992.
12. 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.