Active Rays: Polar–transformed Active Contours for Real–Time Contour Tracking

J. Denzler and H. Niemann

Lehrstuhl für Mustererkennung
Universität Erlangen/Nürnberg
D-91058 Erlangen, Germany

Abstract

In this paper we describe a new approach to contour extraction and tracking, which is based on the principles of active contour models and overcomes its shortcomings. We formally introduce active rays*, describe the contour extraction as an energy minimization problem and discuss what active contours and active rays have in common.*

The main difference is that for active rays a unique ordering of the contour elements in the 2D image plane is given, which cannot be found for active contours. This is advantageous for predicting the contour elements' position and prevents crossings in the contour. Furthermore, another advantage is that instead of an energy minimization in the 2D image plane the minimization is reduced to a 1D search problem. The approach also shows any–time behavior which is important with respect to real–time applications. Finally, the method allows for the management of multiple hypotheses of the object's boundary. This is an important aspect if concave contours shall be tracked.

Results on real image sequences (tracking a toy train in a laboratory scene, tracking pedestrians in an outdoor scene) show the suitability of this approach for real–time object tracking in a closed loop between image acquisition and camera movement. The contour tracking can be done within the image frame rate (25 fps) on standard Unix workstations (HP 735) without any specialized hardware.

1 Introduction

In the past years real–time object tracking in a closed loop of image acquisition and camera movement has become more and more important. Real–time object tracking algorithms are applied to the area of autonomous mobile systems [10], service and cleaning robots and surveillance systems [3]. The typical environment of such systems consists of a dynamically changing world due to motion and actions of objects in the world and due to the movement of the system itself. Thus, no offline processing of the image data is possible. The results of a motion tracking module must be available in time, to suitably react on events in the world. One example are service robots, which must track moving people in hospitals to avoid collisions with them. The tracking algorithm provides information about moving persons, which is used by another module to decide, whether a person might be an obstacle or not. If the moving object is on the movement path of the service robot, the robot must avoid the collision.

Up to now, many different algorithms have been developed to detect and track motion in image sequences [13]. Some of the work was concentrated on offline processing of prerecorded image sequences [12]. This means, that no interaction in a closed loop of image acquisition and camera movement is possible. Real–time object tracking has been the goal of several researchers. Some of them use a model based approach [8, 9] which is dependent on a specific area of application, like car tracking. The other class of algorithms uses correlation or optical flow based approaches [17], which can be applied without knowledge about the problem domain. In the past realization of such algorithms made use of specialized hardware, like pipelined imaging hardware or transputer systems [8, 15]. For portability reasons it would be advantageous, if such systems could be implemented on general purpose hardware. A rule of thumb says, that the hardware performance doubles every 18 months. Thus, no reimplementation on the faster hardware is necessary. The software only needs to be recompiled and then runs twice as fast.

Some promising results in real–time tracking with-

out specialized hardware have been presented recently [1]. One class of algorithm is the so called active contour model (snake) [11]. What are the reasons, for the suitability of active contours for object tracking without specialized hardware? First, the image needs only be processed in a small area around the snake elements, which results in an enormous reduction of the processed data. Second, active contour models need no distinction between foreground and background objects. The active contour extracts the moving object's contour independently of other moving objects in the scene — assuming that there are no occlusions. This is advantageous if the camera is steered to follow the moving object and thus motion is induced to the whole scene. Finally, active contours can handle changes of the object's contour due to the inherent deformation ability. Thus, changing contours of the moving object due to motion in the 3D world, which are based on a changing view to the object, can be handled.

But active contours have also some disadvantages: No ordering of the elements in the image plane is given, it is hard to implement an *any–time* behavior, crossings can occur in the contour, and the contour normally shrinks in the case of missing external energies. These disadvantages will be discussed in more detail in section 2. They have been the motivation of our work. Starting from the advantages of active contours, we modify the contour representation. We use a reference point within the object's contour and shoot rays in different directions from this reference point $\boldsymbol{m}$ (see Figure 1). On these rays, we look for contour point candidates. This reduces the contour point localization from a 2D to a 1D search on each ray. The coupling of the rays in different direction is done similary to active contours. We introduce an energy term, which describes the internal elasticity of the rays. Now we have an active ray following the naming of active contours. The contour extraction can then be described as an energy minimization problem. We show in the following, that all optimization problems are reduced to 1D search problems, and that no crossings can occur because a unique ordering of the contour points in the image plane is given. This is important, if the locations of the contour points shall be estimated during a prediction step. Finally, the contour extraction shows *any–time* behavior. This means, that after an initialization step, the iterative algorithm for extracting the contour of the moving object provides a contour representations after each iteration step. The accuracy of the contour representations grows with each iteration step. Thus, the iterative procedure can be stopped at any time, depending on the available computation time or the required accuracy of the contour extraction. This is a important aspect for a real–time system, because the so called *in–time* constraint can be satisfied. The experimental part will prove, that the contour extraction is robust and can be done within the image frame rate without any specialized hardware. The contour extraction time varies between 9 msec and 38 msec depending on the chosen accuracy. We will prove the robustness of this new approach for laboratory scenes (tracking a moving toy train) and outdoor scenes (tracking moving pedestrians).

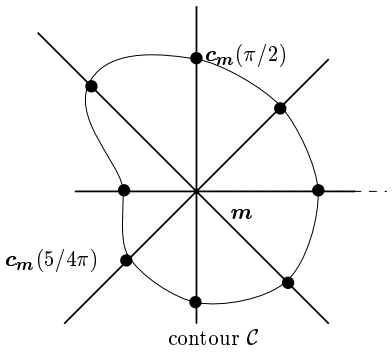

Figure 1: Representation of a contour by active rays

The paper is structured as follows. Section 2 first summarizes the theory of active contours. This theory is used as a motivation to introduce the new method of active rays. We present an energy description, formulate the contour extracting as a energy minimization problem and present an any–time realization of the contour extraction process. In section 3 we go into detail for the formal description of active rays. We formally derive an energy term, starting with the energy description of active contours. With this, we can show, that active rays have the same behavior as active contours. We then simplify the energy term, to avoid the shrinking behavior which can be observed for active contours. In section 4 we present results for the proposed algorithm which show that tracking can be done in real–time on standard Unix workstations without specialized hardware. Section 5 summarizes and discusses the results of this paper.

2 From Active Contours to Active Rays

2.1 A Short Remark on Active Contours

Active contours have been widely used in computer vision in the past eight years, especially in contour segmentation and tracking [16, 2, 14, 4]. An active contour is a parametric function

$$\boldsymbol{c}(s) = (x(s), y(s))^T \in \mathbb{R}^2, \quad s \in [0,1] \tag{1}$$

defined in the (x, y) image plane of an image $f(x, y)$. For closed contours one gets $\boldsymbol{c}(0) = \boldsymbol{c}(1)$. Each snake element $\boldsymbol{c}(s)$ has an energy $E(\boldsymbol{c}(s))$

$$E(\boldsymbol{c}(s)) = E_i(\boldsymbol{c}(s)) + E_e(\boldsymbol{c}(s)), \tag{2}$$

with

$$E_i(\boldsymbol{c}(s)) = \frac{1}{2}\left(\alpha(s)\left|\frac{\partial}{\partial s}\boldsymbol{c}(s)\right|^2 + \beta(s)\left|\frac{\partial^2}{\partial s^2}\boldsymbol{c}(s)\right|^2\right) \tag{3}$$

being the internal energy, and

$$E_e(\boldsymbol{c}(s)) = -\left|G_\sigma \nabla f(\boldsymbol{c}(s))\right|^2. \tag{4}$$

being the external or image energy, smoothed with a Gaussian filter G_σ with variance σ. The active contour has a total energy E

$$E = \int_0^1 E(\boldsymbol{c}(s))ds = \int_0^1 \{E_i(\boldsymbol{c}(s)) + E_e(\boldsymbol{c}(s))\}\, ds. \tag{5}$$

During the contour extraction one looks for a parametric function $\boldsymbol{c}(s)$ which minimizes (5). This is mostly done in the literature by solving the Euler–Lagrange differential equations [11], by the dynamic programming [16], or by the Greedy–Algorithm [18].

In practical applications several problems occur. First, during the energy minimization crossings in the contour may occur [19]. This is shown in Figure 2. These crossings are a serious problem, if features computed from the contour (for example, the center of gravity) are used to track the contour. Of course, such crossings can be detected, but only at the expense of computation time. Second, the snake elements might move around the contour, because they are not fixed at geometric features of the object (see Figure 2). This is a problem for a prediction step, which tries to estimate the motion of the contour in the 2D image plane. If the snake elements move around the contour, a wrong motion will be estimated. Another problem which has often been mentioned is the tendency of the contour to shrink, in the case of missing external forces [18]. This might happen, if parts of the object's contour are weak. As a result the object is lost. Finally, it is hard to implement an any time behavior which would be of great advantage for a real–time application.

The reasons for the crossings in the contour and the movement of the elements around the contour can be understood by looking at the ordering of the active contour elements in the images plane. Due to the definition as a parametric function in $\mathbb{R}^2$, a unique ordering is only given along the contour but not in the image plane. Thus, if we can force an ordering in the image plane, these two problems can be fixed.

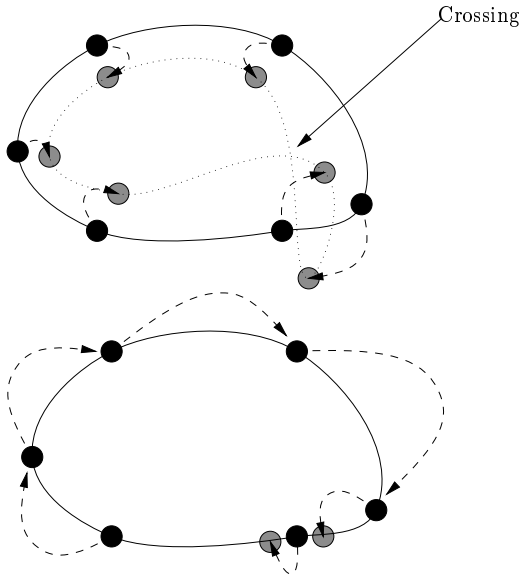

Figure 2: Two main problems of active contours during tracking resulting from the missing ordering in the 2D image plane: Crossings in the 2D contour may occur (top). The snake elements are not fixed at logical features on the object's contour, but they may move around the contour as they like (bottom).

2.2 Active Rays: Energy Description

In the last section the missing ordering in the image plane has been worked out as one reason for some of the problems with active contours. This mean, that we have to introduce a unique ordering in the image plane. For this, we define a reference point $\boldsymbol{m} = (x_m, y_m)^T$, which has to lie within the image contour. An active

ray $\varrho_{\boldsymbol{m}}(\phi,\lambda)$ is defined on the image plane (x,y) as a 1D function depending on those gray values $f(x,y)$ of the image, which are on a straight line from the image point $\boldsymbol{m}$ in direction ϕ

$$\varrho_{\boldsymbol{m}}(\phi,\lambda) = f(x_m + \lambda\cos(\phi), y_m + \lambda\sin(\phi)), \qquad (6)$$

with $0 \leq \lambda \leq n_\phi$, where n_ϕ is given by the image size. In the following we only look at convex contours. Concave contours will be mentioned later. Now, we can identify

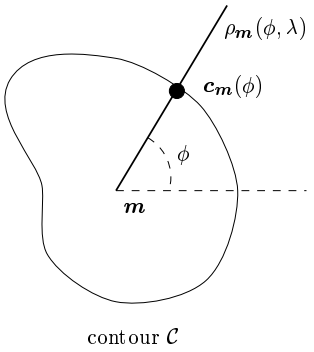

Figure 3: Representation of a contour point by active rays

a point of the contour by the parameter $\lambda^*(\phi) \geq 0$

$$\lambda^*(\phi) \quad = \underset{\lambda}{\operatorname{argmin}} \left(- \left| \frac{\partial}{\partial\lambda} \varrho_{\boldsymbol{m}}(\phi,\lambda) \right|^2 \right), \qquad (7)$$

with $0 \leq \phi < 2\pi$. The step, which lead to (7), is motivated by the assumption, that an edge in 2D can also be found by a gradient search in the corresponding 1D signal. Of course, edges which are in the direction ϕ from the reference point cannot be found on the ray $\varrho_{\boldsymbol{m}}(\phi,\lambda)$. The experiments in section 4 will show, that this case is not relevant in pratice. Having the optimal value for $\lambda^*(\phi)$ the contour point $\boldsymbol{c_m}(\phi)$ in the image plane can easily be computed by

$$\boldsymbol{c_m}(\phi) = (x_m + \lambda^*(\phi)\cos(\phi), y_m + \lambda^*(\phi)\sin(\phi)), \qquad (8)$$

with $0 \leq \phi < 2\pi$. What are the results of this new representation up to now?

1. The ordering in the image plane is given by the angle ϕ, i.e. we always know where the contour point can be found, which corresponds to the direction ϕ_n. For this we only have to look from the reference point in direction ϕ_n. Thus, no crossings can occur in the contour.
2. Using (8) we get the same representation of the contour as for active contour, namely the representation of the contour by the border of the contour.
3. The most important aspect, especially for real–time applications, is the reduction of the contour point search from the 2D image plane to a 1D signal. This reduces the computation time, which will be shown in the experimental part of this paper.

Summarizing the approach, we shoot from one given reference point in different directions ϕ rays, on which a contour point candidate is searched for. In Figure 4 the extracted contour of an object and in Figure 5 the function $\lambda^*(\phi)$ are shown. One can observe, that the function $\lambda^*(\phi)$ is smooth for the angles which corresponds to the correctly extracted contour $(0 - 4/3\pi)$. Then, an error can be seen, both in the extracted contour and in the function $\lambda^*(\phi)$. For $\phi \in [4/3\pi, 3/2\pi\,[$ the function is not smooth, because a wrong contour part has been extracted. This is no surprise. Looking at equation (7) one can see, that up to now, the contour points are calculated without taking into account neighboring contour elements. Thus, we need to introduce some linkage between neighboring contour points to take into consideration that normally contours are coherent in space, i.e. that contours are smooth. A usual approach to connect neighboring contour points together is to introduce an internal energy similar to the active contour approach.

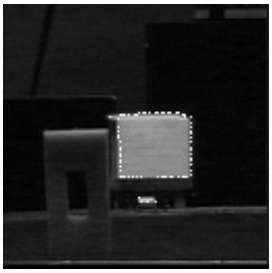

Figure 4: 2D contour extracted by active rays

An internal energy which handles the above mentioned demands is

$$E_i(\boldsymbol{c_m}(\phi)) \quad := \quad \frac{\alpha(\phi)\left|\frac{d}{d\phi}\lambda(\phi)\right|^2 + \beta(\phi)\left|\frac{d^2}{d\phi^2}\lambda(\phi)\right|^2}{2}. \qquad (9)$$

In the next section we will show how this energy can be derived. One important aspect of this internal energy term is, that this energy also depends only on a

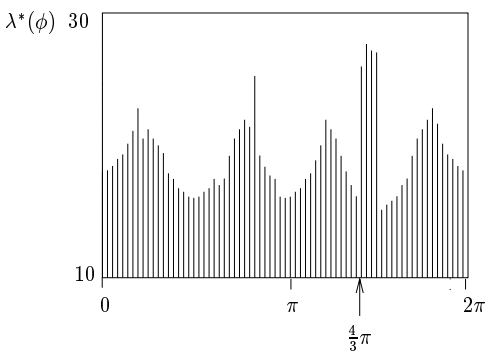

Figure 5: 1D function λ^* of the corresponding 2D contour of Figure 4.

1D function, in contrast to active contours, where the internal energy depends on a 2D function. This results again in a reduction of the complexity of the following optimization algorithms.

Now we have an energy, which describes contour point candidates for each ray and an energy, which connects the rays to get a smooth contour. Similar to active contours we define a total energy E

$$E = \int_0^{2\pi} [E_i(\lambda(\phi)) + E_e(\lambda(\phi))]\, d\phi. \quad (10)$$

The contour extraction can then be described as an energy minimization problem. Using the variational approach the Euler–Lagrange differential equation

$$\alpha(\phi)\frac{d^2}{d\phi^2}\lambda(\phi) - \beta(\phi)\frac{d^4}{d\phi^4}\lambda(\phi) + \frac{d}{d\lambda}\left|\frac{d}{d\lambda}\varrho_{\boldsymbol{m}}(\phi,\lambda)\right|^2 = 0$$

must be solved. Again, this differential equation depends on a 1D function, in contrast to the same differential equation for active contours.

Before we stress the management of multiple hypotheses of contour points on one ray, some remarks must be done regarding the reference point $\boldsymbol{m}$. As already mentioned, this point must lie inside the object contour, but the position may be arbitrary. For a prediction step, a unique position would be of great advantage. Thus, we always choose the center of gravity of the contour

$$\boldsymbol{m} = 1/2\pi \int_0^{2\pi} \boldsymbol{c}_{\boldsymbol{m}}(\phi)\, d\phi \quad (11)$$

for the reference point. If this equation does not hold for an actual reference point and an extracted contour,

INIT:
delta = 2π, shootRay(0), shootRay(π)
ITERATE:
delta = delta/2, angle = 0
WHILE angle less than 2π
shootRay(angle+delta/2)
angle=angle+delta

Figure 6: Any–time algorithm for contour extraction

we update the reference point using equation (11), and restart the contour extraction with the updated reference point's position.

2.3 Discretization: Any–Time Behavior

Up to now we have derived the theory of active rays for the continuous case. By applying active rays for contour extraction in images we have to go to the discrete case. Two approaches for the discretization are possible: fixed sampling rate $\triangle\phi$ for the angle ϕ or the already mentioned any–time behavior. The later one will be discussed in the following.

The representation of active rays allows for an dynamically increasing representation accuracy of the contour. After an initialization step for each iteration we can get a more accurate contour. But we also might stop because after the initialization step we already have a representation of the contour. This representation increases in accuracy for each iteration. If there is only a small amount of time, for example for fast moving objects or the synchronously tracking of several objects, we can stop the iterative procedure after a few iteration steps. If there is more time we can increase the iteration steps and get a more accurate contour representation. Finally, this procedure might be steered by the distance of the contour elements to neighboring contour elements. If the distance between neighboring contour elements corresponding to ϕ_n and ϕ_{n+1} is large, then is would be useful to add one extra ray between the angles ϕ_n and ϕ_{n+1} to get a better approximation of the contour between the angles ϕ_n and ϕ_{n+1}. The algorithm is summerized in Figure 6.

In the experiments, a sampling step size $\triangle\phi = 2\pi/15$ and $\triangle\phi = \pi/5$ have been prooven to be the best values for a wide range of different objects. Increasing $\triangle\phi$ results in a less accurate contour, while decreasing $\triangle\phi$ increases the computation time. A systematic evaluation of different $\triangle\phi$ has been done in [5] for a laboratory scene, based on the difference of the true center of gravity of the moving object and the center of gravity of the

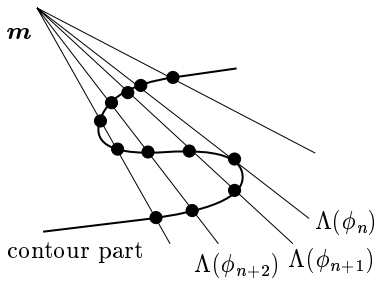

Figure 7: The principle of multiple hypotheses on one ray

extracted contour as well as based on the quality of the complete system (see Section 4.1).

2.4 Extraction of Concave Contours

For extracting concave contours more than one contour point might be found on one ray. Then, multiple hypotheses of the contour points should be handled. This cannot be done for active contours, but can be done for active rays as shown in the following.

We use in the following the abbreviation H for

$$\mathrm{H} = \left(- \left| \frac{\partial}{\partial \lambda} \varrho_{\boldsymbol{m}}(\phi, \lambda) \right|^2 \right). \tag{12}$$

Then, instead of calculating only one contour point candidate for each ray, we can define for each ray in direction ϕ a set $\Lambda(\phi)$

$$\Lambda(\phi) = \left\{ \lambda_k(\phi) \middle| \lambda_k(\phi) = \underset{\lambda, \lambda \neq \lambda_l, l<k}{\operatorname{argmin}} \mathrm{H}, \quad 0 \leq k < i \right\} \tag{13}$$

of i possible solutions for the contour instead of one single contour element. Now, multiple boundary elements lying on one ray can be handled. This is clarified in Figure 7. Now, we have to modify the internal energy of an active ray, because the energy of one ray depends on the hypotheses on this ray and the neighboring rays. What is a good, that means minimal energy, for an active rays with several hypotheses? Instead of having for one contour point a smooth curvature, each hypotheses of this rays should have a corresponding contour point on the neighboring rays. An energy terms which takes this into account is in the discrete

$$E_{i_1}(\Lambda(\phi_n)) = \sum_{\lambda_j \in \Lambda(\phi_n)} \min_{\lambda_k \in \Lambda(\phi_{n+1})} (\lambda_j - \lambda_k)^2 \tag{14}$$

Additionally, large distances between hypotheses on one rays should be preferred. This can be handled by

$$E_{i_2}(\Lambda(\phi_n)) = - \sum_{\lambda_j \in \Lambda(\phi_n)} \sum_{\lambda_k \in \Lambda(\phi_n)} (\lambda_j - \lambda_k)^2, \tag{15}$$

which must me minimized to get large distances between the hypotheses. Now, we get the new term

$$E_i(\Lambda(\phi_n)) = E_{i_1}(\Lambda(\phi_n)) + E_{i_2}(\Lambda(\phi_n)) \tag{16}$$

for the internal energy of an active rays with several hypotheses on each ray. Of course, E_{i_1} and E_{i_2} can be weighted differently. For the external energy we define

$$E_e(\Lambda(\phi_n)) = \sum_{\lambda_j \in \Lambda(\phi_n)} \left(- \left| \frac{\partial}{\partial \lambda} \varrho_{\boldsymbol{m}}(\phi_n, \lambda_j) \right|^2 \right) \tag{17}$$

It is worth noting, that the energy term (14) corresponds only to the first order internal energy of (9).

As already mentioned, in the literature there exist several solutions for the energy minimization problem presented in the previous section. Besides the solution of the Euler–Lagrange differential equation, the dynamic programming treads the optimization as a discrete search problem. Because of the discrete formulation of the multiple hypotheses energies the dynamic programming is better suited for minimizing sum of (16) and (17).

3 Formal Description

In this section we formally derive the internal energy term of the active rays, which has already used in the section 2.2. We can then also explain the tendency of the snakes to shrink and will fix this problem by modifying the internal energy of active contours.

It can be easily seen that the same contour can be represented both with active contours and active rays. Without any loss of generality we assume that the active contour element $\boldsymbol{c}(0)$ corresponds to the active ray element $\boldsymbol{c_m}(0)$. Then the relation

$$\boldsymbol{c}(s) = \boldsymbol{c_m}(\phi), \tag{18}$$

is valid, with the variable substitution $\phi = 2\pi s$.

In the previous section we have looked for an energy term describing some kind of smoothness of a contour. The internal energy of an active contour has been proven to be a good energy description. Thus, we will take exactly this energy term, but substituting the contour representation of snakes by this of active rays. We then get for the internal energy,

$$E_i(\boldsymbol{c_m}(\phi)) = \frac{\alpha(\phi) \left| \frac{d}{d\phi} \boldsymbol{c_m}(\phi) \right|^2 + \beta(\phi) \left| \frac{d^2}{d\phi^2} \boldsymbol{c_m}(\phi) \right|^2}{2}. \tag{19}$$

Now substituting (8) in (19) we get

$$\left(\frac{1}{2\pi}\right)\frac{\partial}{\partial s}\boldsymbol{c_m}(2\pi s) = \frac{\partial}{\partial\phi}\boldsymbol{c_m}(\phi) =$$
$$= \begin{pmatrix} \lambda'(\phi)\cos(\phi) - \lambda(\phi)\sin(\phi) \\ \lambda'(\phi)\sin(\phi) - \lambda(\phi)\cos(\phi) \end{pmatrix} \quad (20)$$

with $\lambda'(\phi) = \frac{\partial}{\partial\phi}\lambda(\phi)$. Similary, we get

$$\left(\frac{1}{2\pi}\right)^2\frac{\partial^2}{\partial s^2}\boldsymbol{c_m}(2\pi s) = \frac{\partial^2}{\partial\phi^2}\boldsymbol{c_m}(\phi)$$

$$= \begin{pmatrix} \lambda''(\phi)\cos(\phi) - 2\lambda'(\phi)\sin(\phi) - \lambda(\phi)\cos(\phi) \\ \lambda''(\phi)\sin(\phi) - 2\lambda'(\phi)\cos(\phi) - \lambda(\phi)\sin(\phi) \end{pmatrix} \quad (21)$$

with $\lambda''(\phi) = \frac{\partial^2}{\partial\phi^2}\lambda(\phi)$. Now one can directly compute

$$\left(\frac{1}{2\pi}\right)^2\left|\frac{\partial}{\partial\phi}\boldsymbol{c_m}(\phi)\right|^2 =$$
$$= \Big[\underbrace{(\lambda(\phi))^2}_{\text{distance term}} + \underbrace{(\lambda'(\phi))^2}_{\text{smoothing term}} \Big] \quad (22)$$

and

$$\left(\frac{1}{2\pi}\right)^4\left|\frac{\partial^2}{\partial\phi^2}\boldsymbol{c_m}(\phi)\right|^2 =$$
$$= \underbrace{4\,(\lambda'(\phi))^2 + (\lambda''(\phi))^2}_{\text{smoothing term}} \underbrace{- 2\lambda''(\phi)\lambda(\phi)}_{\text{mixed term}} + \underbrace{(\lambda(\phi))^2}_{\text{distance term}} \quad (23)$$

and we get for the internal energy $E_i(\boldsymbol{c_m}(\phi))$ of the active ray

$$E_i(\boldsymbol{c_m}(\phi)) = \frac{1}{2}\Big\{\alpha(\phi)(2\pi)^2\left[(\lambda(\phi))^2 + (\lambda'(\phi))^2\right] +$$
$$+ \beta(\phi)(2\pi)^4\left[4\,(\lambda'(\phi))^2 + \right. \quad (24)$$
$$\left. + (\lambda''(\phi))^2 - 2\lambda''(\phi)\lambda(\phi) + (\lambda(\phi))^2\right]\Big\}$$

One can see two distance terms in this energy term. Thus, during an energy minimization small distances to the reference point are preferred. This is exactly the behavior of active contours, which shrink to one point in the case of missing external energy. This is not easy to see by the energy description for active contours. In contrast to this, the energy description of active rays makes this behavior obvious. Using exactly the energy term (25) active rays will show the same abilities as active contours, with the exception, that all 2D optimization steps are reduced to 1D. We have already noted, that the shrinking behavior of active contours is not advantageous. Thus, if we neglect the distance term in the internal energy we fix this problem. Additionally it can be shown, that the mixed term does not influence the solution of the Euler–Lagrange differential equation. So, this term can also neglected. As a result one gets the term for the internal energy, as already proposed in equation (9).

4 Experiments and Results

4.1 Experimental Environment

We have chosen two different experimental environments to show the applicability of active rays: pedestrian tracking and tracking a toy train in a laboratory scene. For the tracking of pedestrians and the toy train, a pan/tilt camera devices looks at the scene.

The system for object tracking runs in two stages: an *initialization stage*, where the motion detection module detects motion in the scene, and a *tracking stage*. Motion detection is done assuming a static camera and computing the difference image between consecutive frames at a 128×128 image resolution. After thresholding and smoothing operations (morphological operators "opening" and "closing") we get binary regions, in which changes in gray values occurred. These changes are assumed to be caused by moving objects. The center of the largest binary region is taken as initialization of the reference point of the active rays. If we observe too many small regions or one very large region, the threshold for computation of the binary image has been set wrong regarding the noise conditions. Thus, we reject the result, increase or decrease the threshold and start the motion detection again. As a result, the threshold is automatically adjusted to the noise conditions in the scene.

After selection of one region and the corresponding center of gravity (taken as initial reference point $\boldsymbol{m}$, compare Section 2.2) the contour extraction starts. As a result we get the new reference point, which is the center of gravity $\boldsymbol{m}$ of the moving object's contour. This positional information is used by a camera control unit, to steer the camera. This is done by a proportional controller to keep the center of the object's contour in the middle of the image. This closes the loop between image acquisition and camera control.

During tracking an attention modules watches over the whole process. This modules computes features for the extracted contour, for example the x– and y–moment of the contour. Based on rapid changes of the features errors are in the contour extraction and

tracking are detected. In this case, the attention module stops the pan/tilt camera, and switches the system back from the tracking stage to the initialization stage. Then, the detection of moving objects starts again.

Due to lack of space, only a short overview of the system could be presented. A more detailed description of this real–time system for tracking moving objects — though using active contours for tracking — can be found in [6]. Experiments, where the camera is mounted on a moving car, can be found in [7].

All algorithms are implemented on Unix–Workstations (SGI Onyx, 2 × R10000) in an object oriented programming language. The frame grabbing is done by a SIRIUS video board, which does no preprocessing. The complete system also runs on a HP (735/99) but with reduced speed due to the moderate frame grabbing rate of an HP RasterOps frame grabber.

4.2 Tracking with Active Rays: Results

Experiments for extracting and tracking a contour with active rays have been conducted. No prediction step has been applied yet. This will be done in our future work. Some ideas together with more experiments (tracking of cars) can be found in [5].

In Figure 8 – 10 three different image sequences taken during a real–time experiment can be seen. In Figure 8

Figure 8: Sequence 1: Tracking a pedestrian with active rays. (images 4, 24, 44, 64, 84, 104 of a sequence taken during a real–time experiment). The sampling step size $\triangle\phi$ is $2\pi/15$.

a moving person is correctly tracked, although the contrast to the background is low. A worse result can be seen in Figure 9. A person approaching the camera is correctly tracked, until the contrast is very low and

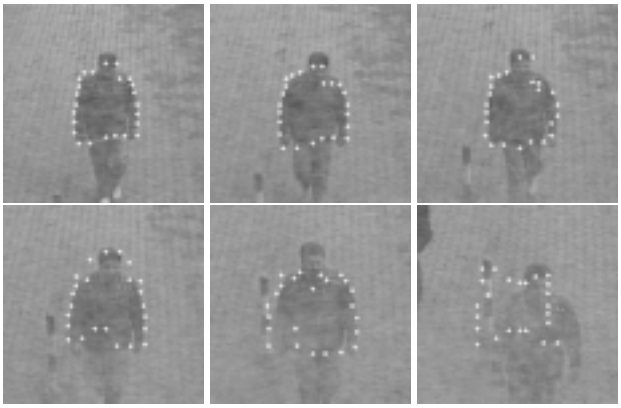

Figure 9: Sequence 2: Tracking a pedestrian approaching the camera (images 19, 29, 39, 49, 59, 69 of a sequence taken during a real–time experiment). The sampling step size $\triangle\phi$ is $2\pi/15$.

another minimum in the energy corresponding to the background object (the stick) is reached. Then, the object is lost. This is a problem of a data driven approach without prediction. To improve the approach we will add a prediction step in our future work. Another good result can be seen in Figure 10. One can also see, that the extracted contour is very accurate. As soon as the pedestrians moves into the sunshine the active ray is attracted by the strong edges between sun and shadow, and the object is lost.

To judge the results for pedestrians tracking under different weather conditions, we have recorded the camera signal during the experiments on a video tape. This video tape has been evaluated by visual inspection, to measure the amount of time, where tracking works well or errors occur. Results are shortly summarized in Table 1. This shows the robustness of the contour based approach regarding noise. The rates for correct motion detection are less accurate due to noise sensitivity of the difference image motion detection algorithm. As already mentioned the approach is sensitive to weak object contours.

In Table 2 the computation time on a HP (735/99) for different sampling step sizes $\triangle\phi$ can be seen. Even for a very dense sampling ($\triangle\phi = \pi/180$) the contour extraction can be done within the image frame rate.

Finally, Figure 11 shows some results for the laboratory scene. During a 30 minute experiment, which corresponds to 45000 images, the moving toy train has been tracked without any error, even during occlusions.

Figure 10: Sequence 3: Tracking a pedestrian approaching the camera (images 1, 21, 41, 71, 101, 131 of a sequence taken during a real–time experiment). The sampling step size $\triangle\phi$ is $2\pi/15$.

weather	**total time**	**# persons**	**time of tracking**	**# tracked persons (time)**
snow	120	226	21	101 (11)
cloudy	90	128	14	93 (11)
sunshine	90	152	13	102 (12)
total	300	506	48	296 (34)

Table 1: Results for pedestrian tracking under different weather conditions (success rate: 70 %). The time is given in minutes.

The train moves at a speed of 40 cm/sec at a distance of 1.5 to 2.5 m to the camera. This corresponds to a displacement between consecutive images of 6–8 pixels at a resolution of 128×128 pixels. The same experimental environment has been taken for tracking with active contours. This shows the improvements by the new approach. Results can be seen in Figure 12. In this case, the maximum speed of the toy train for comparable tracking results has been 2.4 cm/sec, which prooves the expected reduction of computation time for active rays.

For more detailed evaluations of the real–time tracking system, in which the approach of active rays is included, we have to refer to [5] for the laboratory scene as well as for the natural one.

$\triangle\phi$	time/image (msec)
$\pi/180$	38
$\pi/36$	19
$\pi/18$	12
$\pi/9$	9

Table 2: Computation time for one image for different sampling step sizes $\triangle\phi$.

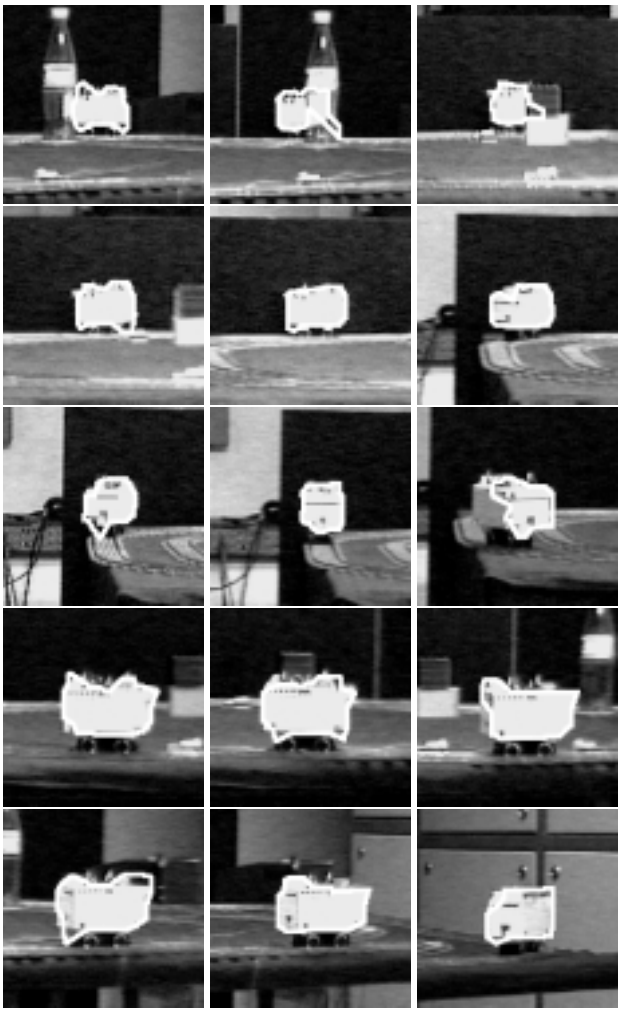

Figure 11: Sequence 4: Laboratory scene with partial occlusions and random changes of direction and speed of the toy train during tracking. Every 5th image of a sequence taken during a real–time experiment is shown. The sampling step size $\triangle\phi$ is $\pi/5$.

5 Conclusion and Future Work

In our contribution we have presented a new approach to contour tracking, called active rays. The basic ideas come from active contours, which have been proven to be a promising approach to data driven real–time contour tracking.

Active rays have the following advantages over active contour models:

- For active rays an ordering in the image plane is given by a reference point $\boldsymbol{m}$ and an angle ϕ. Thus no crossings occur and predicting the position of the contour elements is possible.
- All optimization problems are reduced to 1D search problems.
- Active rays provide a mechanism to select the required accuracy of the contour approximation. This leads to an any–time behavior, which is an important aspect of real–time applications.
- Active rays provide a mechanism to manage multiple hypotheses. An energy term has been presented, which allows for the extraction of concave contour parts within a energy minimization framework.

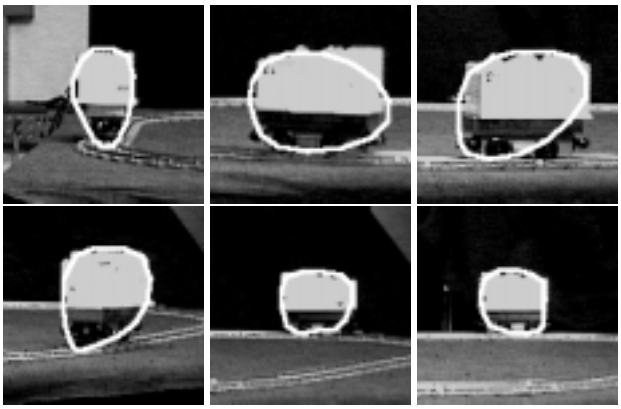

Figure 12: Sequence 5: Results for tracking the toy train with active contours (20 contour points). The maximum speed, where robust tracking can be performed, has been 2.4 cm/sec.

We have presented a formal description of active rays and an energy formulation for the contour extraction. In addition we have shown the common parts of active contours and active rays. Using the dynamic programming for the energy minimization, the extraction of concave contours is possible. The experiments have proven that this new approach is well suited for an accurate contour extraction and tracking in real–time. For this no specialized hardware is necessary. The information of the tracking algorithm, can be used as input for another module, for example navigation or obstacle avoidance.

We have focussed on a real–time application in a closed loop between image acquisition and reaction — in our contribution the pan/tilt movement of a camera. The approach of active rays as well as active contours (see for example [7]) can also be applied to multiple object tracking by using one reference point for each object. In this case either the camera movement and the selection of the moving objects need to be optimized to synchronously track multiple, or the camera needs be static during tracking, too. Both problems are very interesting and will be investigated in our future work.

In our future work we will focus on a second order energy term for multiple contour points on one ray. Additionally, we have first results for texture based energies on 1D signals instead of the gradient in (7). Then also textured object can be tracked. The combination of active rays with a prediction step should also improve the tracking results.

References

[1] M. Armstrong and A. Zisserman. Robust object tracking. In *Second Asian Conference on Computer Vision*, pages I/58–I/62, Singapore, 1995.

[2] 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.

[3] R. Curwen, A. Blake, and A. Zisserman. Real–time visual tracking for surveillance and path planning. In G. Sandini, editor, *Computer Vision - ECCV 92*, pages 879–883, Berlin, Heidelberg, New York, London, 1992. Lecture Notes in Computer Science.

[4] P. Delagnes, J. Benois, and D. Barba. Active contours approach to object tracking in image sequences with complex background. *Pattern Recognition Letters*, 15:171–178, 1995.

[5] J. Denzler. *Active Vision for Real–Time Object Tracking*. Dissertation, Technische Fakultät, Universität Erlangen–Nürnberg, Erlangen, to appear, 1997.

[6] J. Denzler and H. Niemann. Combination of simple vision modules for robust real–time motion tracking. *European Transactions on Telecommunications*, 5(3):275–286, 1995.

[7] J. Denzler and H. Niemann. 3D data driven prediction for active contour models with application to car tracking. In *Machine Vision Application*, pages 204–207, Tokyo, 1996.

[8] E.D. Dickmanns, R. Behringer, C. Brüdigam, D. Dickmanns, F. Thomanek, and V.v. Holt. An all–transputer visual autobahn–autopilot/copilot. In *International Conference on Computer Vision*, pages 608–615, Berlin, 1993.

[9] C. Harris. Geometry from visual motion. In A. Blake and A. Yuille, editors, *Active Vision*, pages 239–262. MIT Press, Cambridge, Massachusetts, London, England, 1992.

[10] E. Huber and D. Kortenkamp. Using stereo vision to pursue moving agents with a mobile robot. In *Proceedings of IEEE Conference on Robotics and Automation*, pages 426–431, 1995.

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

[12] D. Koller, K. Daniilidis, T. Thorhallson, and H. Nagel. Model–based object tracking in traffic scenes. In G. Sandini, editor, *Computer Vision - ECCV 92*, pages 437–452, Berlin, Heidelberg, New York, London, 1992. Lecture Notes in Computer Science.

[13] A. Mitiche and P. Bouthemy. Computation and analysis of image motion: A synopsis of current problems and methods. *International Journal of Computer Vision*, 19(1):29–55, 1996.

[14] K.P. Ngoi and J. Jia. A robust active contour model for natural scene contour extraction with automatic thresholding. In A. Blake, editor, *Computer Vision - ECCV 96*, pages 335–348, Berlin, Heidelberg, New York, London, 1996. Lecture Notes in Computer Science.

[15] 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.

[16] N. Ueda and K. Mase. Tracking moving contours using energy–minimizing elastic contour models. In G. Sandini, editor, *Computer Vision - ECCV 92*, pages 453–457, Berlin, Heidelberg, New York, London, 1992. Lecture Notes in Computer Science.

[17] T. Uhlin, P. Nordlund, A. Maki, and J.O. Eklundh. Towards an active visual observer. In *International Conference on Computer Vision*, pages 679–686, Cambridge, Massachusetts, 1995.

[18] D.J. Williams and M. Shah. A fast algorithm for active contours and curvature estimation. *Computer Vision, Graphics, and Image Processing*, 55(1):14–26, 1992.

[19] N. Yokoya and S. Araki. Splitting contour models based on crossing detection. In *Proceedings of the 1995 Real World Computing Symposium*, pages 29–30, Tokyo, 1995.