An Efficient Combination of 2D and 3D Shape Descriptions for Contour Based Tracking of Moving Objects

J. Denzler, B. Heigl*, H. 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@informatik.uni-erlangen.de

Abstract Tracking the 2D contour of a moving object has widely been used in the past years. So called active contour models have been proven to be a promising approach to real–time tracking of deformable objects. Also tracking 2D contours, which are projections of rigid 3D objects, is reduced to tracking deformable 2D contours. There, the deformations of the contour are caused by the movement in 3D and the changing perspective to the camera.
In this paper a combination of 2D and 3D shape descriptions is presented, which can be applied to the prediction of changes in 2D contours, which are caused by movement in 3D. Only coarse 3D knowledge is provided, which is automatically acquired in a training step. Then, the reconstructed 3D model of the object is used to predict the shape of the 2D contour. Thus, limitations of the contour point search in the image is possible, which reduces the errors in the contour extraction caused by heterogenous background.
The experimental part shows, that the proposed combination of 2D and 3D shape descriptions is efficient and accurate with respect to real–time contour extraction and tracking.

1 Introduction

In the past years a new framework has been established for representing a deformable object by its contour. Active contours [11] and related models have been developed and have been widely used in computer vision, for example for medical imaging, vision aided speech recognition (lip reading), segmentation and tracking [3, 14, 16].

Although active contour models have been used for tracking moving objects, it is an open question how changes in the contour during tracking can be predicted. Using the data driven approach of active contours usually no a priori knowledge of the object is available. Only a few mechanisms have been included providing the possibility for predicting the deformation of the object's contour. For this, most researchers concentrate on the prediction of 2D changes without taking into account the motion of the object

* This work was partially funded by the German Research Foundation (DFG) under grant number SFB 182. Only the authors are responsible for the contents.

in 3D. [19] presents a Kalman–Snake combining the principles of active contours and prediction by a Kalman–Filter. In [1] the motion of the contour in the 2D image plane is predicted by computing the normal flow at the contour points. In the case of tracking rigid or elastic objects moving in a 2D plane parallel to the image plane some work have been presented which learn the possible deformations of the object and limit the contour point search on certain areas in the image [4, 12].

In the following, we focus on tracking the contours of rigid objects moving in 3D. Then, the deformation of the contour depends on the 3D shape of the object and is caused by the changing view to the camera. Without any prediction, problems arise, when the object moves in front of a heterogeneous background. Strong background edges often define local minima during the contour extraction process. As a result, the active contour is caught by the background and the moving object is lost.

One straightforward approach to handle these problems is to predict the motion of the contour in the 2D image plane. This works very well (see [19]) in the case of tracking a single patch of the object or tracking an object, whose visible patches do not change. In contrary, if the visible parts of the object change, new object edges appear near the object itself (see Figure 1, (a)). Without any coarse knowledge about the 3D shape of the object these new edges cannot be distinguished from appearing background edges which have been covered before by the object (Figure 1, (b)).

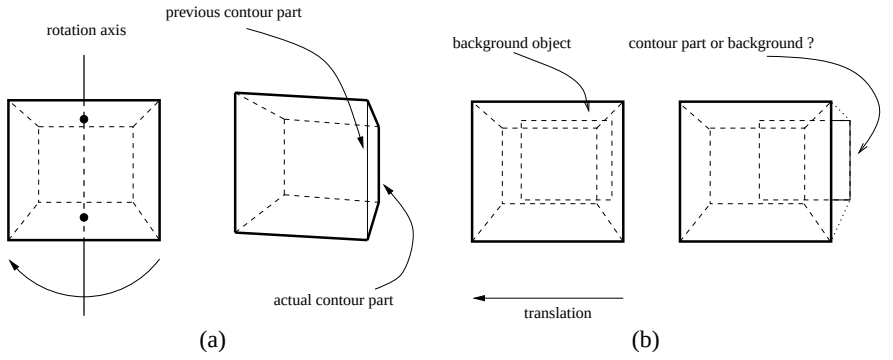


Figure1. Problem of predicting new contour parts: how can an active contour distinguish between a new contour part and an appearing background edge? (a) rotation of object. (b) translation and appearing background object.

How can we get a solution for this problem? On the one hand we do not like to give up the data driven approach allowing a very fast and efficient tracking of moving objects even in real–time [5]. On the other hand, one needs to introduce knowledge about the moving object to increase robustness during tracking. Using knowledge can be a time consuming task which might prevent the use in real–time applications.

In our contribution we concentrate on this trade–off. We present an efficient combination of a 2D data driven contour extraction method and a 3D shape modelling

technique. For 2D modelling of the contour as well as for 3D shape modelling a radial representation is used. Both representations can easily be mapped on each other, i.e. having several 2D views of the object the 3D representation can be built; having the 3D representation the corresponding 2D contour can be predicted. And with a 3D representation and a single 2D contour corresponding to a unique view of the object the position of the object in 3D can roughly be estimated. For both the 2D and 3D case, it is possible to dynamically adjust the accuracy of the representation to the available computation time. This is an important feature for real–time applications. Thus, possible applications for our method exist in the field of autonomous mobile systems, where in different situations different objects need to be tracked: from unknown objects (obstacles) which need to be identified quickly and tracked without knowledge up to known objects in visual grasping tasks, where more time can be spent. Finally, the 3D contour prediction method itself can also be integrated in arbitrary contour tracking algorithms.

In contrast to model based tracking algorithms [9, 10], we are not interested in an accurate 3D representation of the object. Active contours need only a coarse initialization near the contour which should be extracted. In our approach, the model of the object can also be constructed very quickly by presenting a sufficient number of 2D views and applying shape from contour methods. Finally, the experimental part will prove, that the 2D contour extraction, the 3D model construction as well as the prediction can be done in real–time on general purpose hardware.

The paper is structured as follows. In Section 2 we shortly summarize the principles of 2D contour extraction by active rays. Section 3 discusses the basic concepts of shape from contour which is the key idea for our automatic shape reconstruction step. We focus on a radial representation of the object's shape by rays in Section 4, and show how the accuracy of the shape description can be adjusted dynamically. In Section 5 the basic concepts of shape from contour, the 2D contour representation, and 3D shape models are merged together to automatically reconstruct a 3D model by single 2D views. We also show, how the resulting coarse 3D model can be used to predict the changes in the 2D contour. In Section 6, experiments show the improvement for tracking objects moving in 3D. The paper finishes with a discussion in Section 7.

2 A Radial Representation for Contour Extraction

In this section a new approach will be shortly summarized for 2D contour extraction. A detailed description can be found in [6]. Instead of modelling the elastic contour by a parametric function in the 2D image plane [11], a different strategy has been developed.

An initial reference point $\boldsymbol{m} = (x_m, y_m)^T$ inside the contour, which should be extracted, has to be chosen. This is similar to a balloon model [3], where the initial balloon also needs to be inside the contour of the object. In the next step the 2D image $f(x, y)$ is sampled along straight lines — so called *rays* —

$$\varrho_{\boldsymbol{m}}(\phi, \lambda) = f(x_m + \lambda\cos(\phi), y_m + \lambda\sin(\phi)), \tag{1}$$

from the reference point in certain directions. As a result for each ray we get 1D gray value signals which depend on the chosen reference point $\boldsymbol{m}$. Usually, for ϕ the range from $[0, 2\pi[$ is sampled more or less dense, depending on the necessary accuracy of

the extracted contour. For real–time applications, this can be controlled by the available computation time.

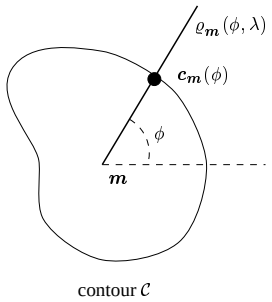


Figure2. Representation of a contour point by active rays

Now, for contour extraction the rays $\varrho_{\boldsymbol{m}}(\phi,\lambda)$ are taken. For each 1D gray value signal, features are computed to identify the position $\lambda^*(\phi) \geq 0$ of the object's boundary on the ray. One possible criterion, which we call external energy, is the gradient

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

of the 1D gray value signal to identify contours by changes in the gray values. Also, more sophisticated energies have been used, for example changes in the variance of the gray values to extract textured regions.

Based on these features for direction ϕ the position $\lambda^*(\phi)$ is identified, which corresponds to the object boundary

$$\boldsymbol{c_m}(\phi) = (x_m + \lambda^*(\phi)\cos(\phi), y_m + \lambda^*(\phi)\sin(\phi)), \tag{3}$$

in the 2D image plane, with $0 \leq \phi < 2\pi$. This works well for convex contours, because then each ray only hits the object contour once. Depending on the position of the reference point, this is also true for some concave contours. In the general case of concave contours, a mechanism has been provided, which allows more than one contour point on each 1D gray value signal. This case will not be discussed in this paper. A detailed description can be found [6]. In contrast to [18], where also a radial representation is proposed, but no energy description for the contour extraction, we force — similar to active contours — the smoothness of the 2D contour $\boldsymbol{c_m}(\phi) \in \mathbb{R}^2$ by defining an internal energy E_i

$$E_i(\boldsymbol{c_m}(\phi)) = \frac{\alpha(\phi)\left|\frac{d}{d\phi}\lambda(\phi)\right|^2 + \beta(\phi)\left|\frac{d^2}{d\phi^2}\lambda(\phi)\right|^2}{2}. \tag{4}$$

The internal energy is based on the distance of the contour point to the reference point. Since the derivative of the distance $\frac{d}{d\phi}\lambda(\phi)$ and the curvature $\frac{d^2}{d\phi^2}\lambda(\phi)$ describe the smoothness of the contour, this energy forces coherence of the contour in the image plane. Now, the contour extraction, i.e. searching for the function $\lambda^*(\phi)$ can be described as a minimization of the total energy E

$$\lambda^*(\phi) = \underset{\lambda}{\operatorname{argmin}} \int_0^{2\pi} \left[\frac{\alpha(\phi)\left|\frac{d}{d\phi}\lambda(\phi)\right|^2 + \beta(\phi)\left|\frac{d^2}{d\phi^2}\lambda(\phi)\right|^2}{2} - \left|\frac{d}{d\lambda}\varrho_{\boldsymbol{m}}(\phi,\lambda)\right|^2 \right] d\phi. \quad (5)$$

This algorithm has been successfully applied to real–time pedestrian tracking in natural scenes [8]. More details of the mathematical derivation can be found in [6]. In this paper we focus on the combination with a 3D prediction step. In the next two sections we summarize the mathematical preliminaries for 3D object modelling and shape from contour. In Section 5 we will show, that in a combination with a similar 3D representation the 2D radial representation has advantages. It is beyond the scope of this paper to make comparisons to other contour based tracking algorithms, especially active contours. Such a discussion can be found elsewhere.

3 Concept of 3D Reconstruction from 2D Contours

We are interested in tracking rigid objects moving in 3D. We focus on contour based methods (see Section 2). Thus, we need to describe formally, how 2D contours of 3D objects are generated. An important term in this context is the silhouette of an object. Since the next section describes how to build an object description out of segmented silhouettes, it is also important to define two further concepts, the visual and outer hull, which give approximations of 3D objects.

Let the object O be represented by a set of points $\boldsymbol{x} \in \mathbb{R}^3$. The points $\boldsymbol{x}_{\mathrm{M}} \subset O$ are constant within the model coordinate system. In the following we assume, that the transformation of a point in model coordinates $\boldsymbol{x}_{\mathrm{M}}$ to camera coordinates $\boldsymbol{x}_{\mathrm{C}}$ is given by the relation $\boldsymbol{x}_{\mathrm{C}} = \boldsymbol{R}_{\mathrm{MC}}\boldsymbol{x}_{\mathrm{M}} + \boldsymbol{t}_{\mathrm{MC}}$. In the case of model reconstruction, the rotation matrix $\boldsymbol{R}_{\mathrm{MC}}$ and the translation vector $\boldsymbol{t}_{\mathrm{MC}}$ are assumed to be known, either by defining the viewing angle in a training stage or by estimating the pose of the object from 2D images.

3.1 Object Silhouettes

A descriptive illustration of an object silhouette is the shadow of an object produced on a plane by a single light spot. Following [13], a definition of a silhouette might be:

> The word *silhouette* indicates the region of a 2D image of an object O which contains the projections of the visible points of O.

The projection of object points to image points can be described by many different models. Generally, the projection function is denoted by $\mathcal{P} : \mathbb{R}^3 \rightarrow \mathbb{R}^2$. Here we only consider the cases of orthogonal and perspective projection, but other projections

are adaptable to the later concepts without big efforts. The backprojection function $\mathcal{P}^{-1}: \mathbb{R}^2 \rightarrow \mathbb{R}^3$, concerning a specified projection $\mathcal{P}$, is defined by

$$\mathcal{P}^{-1} := \boldsymbol{x}_{\mathrm{P}} \rightarrow \{\boldsymbol{x}_{\mathrm{C}} \in \mathbb{R}^3 \,|\, \mathcal{P}(\boldsymbol{x}_{\mathrm{C}}) = \boldsymbol{x}_{\mathrm{P}}\}. \tag{6}$$

This mapping of image points $\boldsymbol{x}_{\mathrm{P}}$ to points in $\boldsymbol{x}_{\mathrm{C}}$ camera coordinates is not unique, because of $\mathcal{P}$ not being injective.

Using these definitions, the silhouette of an object O in the pose $(\boldsymbol{R}_{\mathrm{MC}}, \boldsymbol{t}_{\mathrm{MC}})$ created by projection $\mathcal{P}$ is formally denoted by

$$S_{\boldsymbol{R}_{\mathrm{MC}}, \boldsymbol{t}_{\mathrm{MC}}} = \{\mathcal{P}(\boldsymbol{R}_{\mathrm{MC}}\boldsymbol{x}_{\mathrm{M}} + \boldsymbol{t}_{\mathrm{MC}}) \mid \boldsymbol{x}_{\mathrm{M}} \in O\}. \tag{7}$$

It is assumed, that the silhoutte has no “hole”, therefore it can be represented by its contour. As we know the contour of an object by tracking it using active rays, we have a representation of the object silhouette at any time of the trace.

3.2 Outer and Visual Hull

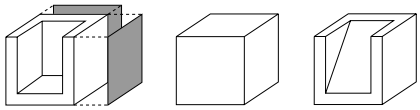


Figure3. Building two silhouettes of an object (a), the outer hull of these silhouettes (b) and the visual hull (c)

Now we examine the problem of reconstructing an object out of its silhouettes. Given n sequentially recorded silhouettes ${}^{t}S_{\boldsymbol{R}_{\mathrm{MC}}(t), \boldsymbol{t}_{\mathrm{MC}}(t)}$, $1 \leq t \leq n$, we define the abbreviation $S(t) = {}^{t}S_{\boldsymbol{R}_{\mathrm{MC}}(t), \boldsymbol{t}_{\mathrm{MC}}(t)}$. The backprojection $C_{\mathrm{M}}(t)$ of the Silhouette $S(t)$ is given by the union of the backprojections of all points $\boldsymbol{x}_{\mathrm{P}} \in S(t)$. Written in model coordinates, we can calculate

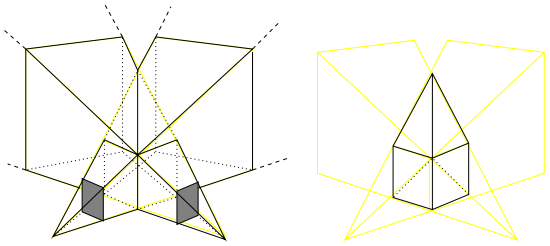
$$C_{\mathrm{M}}(t) = \boldsymbol{R}_{\mathrm{MC}}^{T}(t) \cdot (\mathcal{P}^{-1}(S(t)) - \boldsymbol{t}_{\mathrm{MC}}(t)), \tag{8}$$

where an operation $\circ$ between a set B and an element $\boldsymbol{b}$ is defined as the set of results of the operation between this vector and each element of B: $B \circ \boldsymbol{b} := \{\boldsymbol{x} \circ \boldsymbol{b} \mid \boldsymbol{x} \in B\}$.

Using equation 8, the concept of the outer hull A_{M} is defined in [17] as follows:

$$A_{\mathrm{M}} := \bigcap_{t=t_1}^{t_2} C_{\mathrm{M}}(t) \tag{9}$$

Figure 3 shows an example, comparing the principles of the outer and visual hull. Reconstructing objects by building the outer hull, is known in the literature as so called volume intersection algorithms or as shape from contour techniques. In [13] an overview therefore is given. Figure 4 illustrates this concept. The outer hull is built out of two



(a) (b)

Figure4. Illustration of building the outer hull of two given rectangular silhouettes by intersecting their backprojections. (a): intersection, (b): outer hull

Algorithm to calculate the outer hull of silhouettes	
Parameters: $S(t), R_{MC}(t), t_{MC}(t), 1 \leq t \leq n$	
Initialization: ${}^{0}A_{M} := \mathbb{R}^3$	
FOR all t with $1 \leq t \leq n$	
	Transformation into camera coordinates: ${}^{t-1}A_{C} := R_{MC}(t) \cdot {}^{t-1}A_{M} + t_{MC}(t)$
	Intersection with backprojection of silhouette: ${}^{t}A_{C} := {}^{t-1}A_{C} \cap \mathcal{P}^{-1}(S(t))$
	Transformation into model coordinates: ${}^{t}A_{M} := R_{MC}^{T}(t)({}^{t}A_{C} - t_{MC}(t))$
Return: outer hull ${}^{n}A_{M}$	

Figure5. Algorithm for the iterative calculation of the outer hull

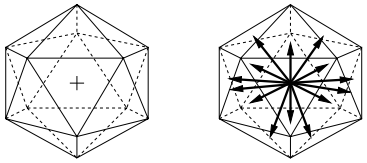
rectangular silhouettes. In the case of central perspective projection, the backprojections of rectangles are pyramids with infinite height. These pyramids are intersected, to build the outer hull.

Joining all possible backprojections of an object, an upper limit for the outer hull is given by the concept called visual hull $H(O)$ of an Object O. It can be described in correspondence to [13] as the unity of all points, whose backprojection of any projection intersects the object O. The following relation holds: $O \subseteq H(O) \subseteq A_{M}$. We now use the outer hull A_{M} as an approximation of the true object O. As already mentioned, we can do this, because in the context of active contours, a coarse initialization of the contour around the true contour is sufficient.

The associative property of the uniting operation allows the construction of the outer hull as it is described in Figure 5. This algorithm has to be applied to the special 3D representation, which is used to model the point set, building the outer hull.

4 3D Radial Object Representation

The idea of representing contours with 2D rays is now extended to the third dimension.



(a) (b)

Figure6. Definition of ray directions as normal vectors on the boundary plains of regular polyhedrons; (a) icosahedron, (b) icosahedron with normal vectors

To avoid confusions, following notations are used to distinguish between the 2D and 3D case: the accent ˜ denotes, that the corresponding symbol belongs to 3D rays, the accent ˆ, that it belongs to the projection of a 3D ray, and without accent, that it belongs to a 2D ray. The 3D ray representation $\tilde{K}$ can then be written as a tuple:

$$\tilde{K} := (\tilde{W},\ \tilde{\boldsymbol{m}},\ \tilde{\lambda} : \tilde{W} \rightarrow \mathbb{R} \cup \infty). \tag{10}$$

$\tilde{W}$ is the set of direction vectors $\boldsymbol{n}$ of the predefined 3D rays with $\|\boldsymbol{n}\| = 1$. The vector $\tilde{\boldsymbol{m}} \in \mathbb{R}^3$ denotes the reference point and the function $\tilde{\lambda}$ returns the length of the ray in direction $\boldsymbol{n}$. The value ∞ means, that this ray length is undefined, yet.

Each ray represents a single point on the surface of the object. If we define the representation continuously, i.e. $\tilde{W} = \{\boldsymbol{x} \in \mathbb{R}^3 \mid \|\boldsymbol{x}\| = 1\}$, the represented point set is equal to $\{\boldsymbol{x} \in \mathbb{R}^3 \mid \boldsymbol{x} = \tilde{\boldsymbol{m}} + k\boldsymbol{n}, 0 \leq k \leq \tilde{\lambda}(\boldsymbol{n}), \boldsymbol{n} \in \tilde{W}\}$.

In the discrete case, the choice of the directions $\tilde{W}$ should be distributed regularly. Moreover the resolution should be changeable dynamically to achieve any-time behavior. If we define the regularity in the sense that neighboring directions shall enclose all the same angle, we are restricted to 20 directions, defined by the normal vectors of the icosahedron, being the regular polyhedron with the most plains (see Figure 6). This is equal to take the 20 corners of the dodecahedron, being the regular polyhedron with the most corners.

In [2] a method is described, to subdivide an icosahedron, so that the error in the regularity is minimized. Taking the 20 normals and the 12 corners of the icosahedron, 60 triangular plains can be defined.

This approximation can be refined by subdividing the triangles into subtriangles, defining three new points bisecting the three sides.

Using this representation, all convex and some concave objects can be described. Like in the 2D case concave objects exist, which cannot be represented. Also holes in the objects can better be approximated using different representations (for example octrees); since we are only interested in the contour and a efficient mapping from 3D to 2D rays and vice versa, this is not relevant for our approach.

5 Application to Contour Based Object Tracking

We have seen, how to build the outer hull of an object from its silhouettes (Section 3). Besides we have designed a 3D representation by 3D rays, similar to the 2D ray representation (Section 4). Using these concepts, it is shown how to create the 3D object model (Section 5.1) and to extract its contour in a given pose (Section 5.2). Section 5.3 describes how to use this model to predict contours during tracking objects.

5.1 3D Model Generation

Now we apply the algorithm described in Figure 5 to the 3D representation developed in Section 4. After each iteration step, the 3D ray model shall represent the outer hull, which is generated by intersecting the backprojections of the extracted silhouettes.

The goal is to represent the outer hull A_M by the 3D ray representation $\tilde{K}_M = (W_M, 0_3, \tilde{\lambda}_M)$. The reference point is set to 0_3 without any loss of generality.

According to Figure 5, ${}^0A_M$ has to be set to $\mathbb{R}^3$ at the beginning. This is done in ${}^0\tilde{K}_M$ by assigning ${}^0\tilde{\lambda}_M(\boldsymbol{n}_M) = \infty$. In a second step, the set ${}^tA_M$ — using model coordinates — has to be transformed to camera coordinates. This is done by transforming $\tilde{K}_M$ to $\tilde{K}_C$ following:

$$\begin{aligned} {}^t\tilde{K}_C &= (W_C,\ \tilde{\boldsymbol{m}}_C,\ \tilde{\lambda}_C),\ \text{with} \\ \tilde{W}_C &= \boldsymbol{R}_{MC}(t) \cdot \tilde{W}_M, \\ \tilde{\boldsymbol{m}}_C &= \boldsymbol{t}_{MC}(t), \\ \tilde{\lambda}_C(\boldsymbol{n}_C) &= \tilde{\lambda}_M(\boldsymbol{n}_M) \text{ where } \boldsymbol{n}_C = \boldsymbol{R}_{MC} \cdot \boldsymbol{n}_M. \end{aligned} \tag{11}$$

${}^t\tilde{K}_C$ therefore represents the set ${}^tA_C$.

Now, ${}^t\tilde{K}_C$ has to be intersected with the backprojection of the Silhouette $S(t)$, represented by the active contour ${}^tK$. It is mathematically clear, that for an object O, a silhouette S and an arbitrary point $\boldsymbol{p} \in \mathbb{R}^3$, the relation $\mathcal{P}(\boldsymbol{p}) \notin \mathcal{P}(H(O))[= \mathcal{P}(O) = S] \Rightarrow \boldsymbol{p} \notin H(O)$ holds. Besides we know $\mathcal{P}(\boldsymbol{p}) \in S \Leftrightarrow \boldsymbol{p} \in \mathcal{P}^{-1}(S)$. We see, that all points $\boldsymbol{p}$ with $\mathcal{P}(\boldsymbol{p}) \notin S$ have to be removed. Applying this to the ray representation, only those rays are affected, whose projections are not totally contained in the silhouette. These rays have to be cut, so that the ends of their projections lie on the border of the silhouette. Figure 7 shows this operation and the notations used here.

Let us describe this cutting step formally, by first calculating the 2D projection of each of the 3D rays. The starting point of each ray, i.e the reference point $\tilde{\boldsymbol{m}}_C$, is projected to $\mathcal{P}(\tilde{\boldsymbol{m}}_C)$. For each 3D direction $\boldsymbol{n}_C \in \tilde{W}_C$ the corresponding angle $\phi(\boldsymbol{n}_C)$ and the length $\hat{\lambda}(\phi(\boldsymbol{n}_C))$ can be determined by applying the projection function $\mathcal{P}$.

These projected rays have to be intersected with the extracted contour ${}^tK$. Therefore, we overlay ${}^tK$ with the projected rays. This is done by setting the reference point $\boldsymbol{m} := \mathcal{P}(\tilde{\boldsymbol{m}}_C)$, being possible, because $\boldsymbol{m}$ can be chosen arbitrarily.

According to the preliminaries, a unique intercept point of the backprojection of the contour point $\boldsymbol{c}(\phi(\boldsymbol{n}_C)) = \boldsymbol{m} + \lambda(\phi(\boldsymbol{n}_C)) \begin{pmatrix} \cos(\phi(\boldsymbol{n}_C)) \\ \sin(\phi(\boldsymbol{n}_C)) \end{pmatrix}$ with the ray in direction $\boldsymbol{n}_C$ exists. The intersection of a 3D ray with the backprojection of the silhouette S therefore is done by setting the ray length $\tilde{\lambda}_C(\boldsymbol{n}_C)$ to $\rho(\boldsymbol{n}_C)$, if the former ray length was larger.

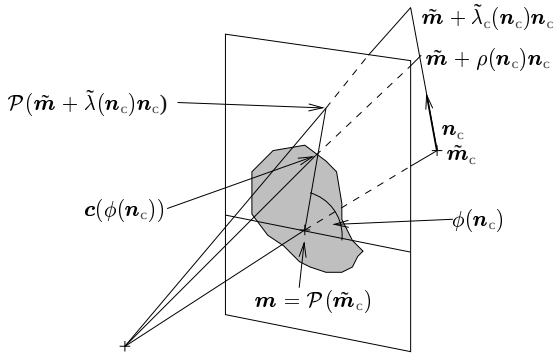


Figure7. Illustration of the cutting algorithm

The value $\rho(\boldsymbol{n}_{\mathrm{C}})$ is defined by following relation: $\tilde{\boldsymbol{m}}_{\mathrm{C}} + \rho(\boldsymbol{n}_{\mathrm{C}})\boldsymbol{n}_{\mathrm{C}} \in \mathcal{P}^{-1}(\boldsymbol{c}(\phi(\boldsymbol{n}_{\mathrm{C}})))$. For the case of parallel or perspective projection, $\rho(\boldsymbol{n}_{\mathrm{C}})$ can be determined easily: $\rho(\boldsymbol{n}_{\mathrm{C}}) = \frac{\tilde{\lambda}(\boldsymbol{n}_{\mathrm{C}})}{\tilde{\lambda}(\phi(\boldsymbol{n}_{\mathrm{C}}))} \cdot \lambda(\phi(\boldsymbol{n}_{\mathrm{C}}))$. Formally, the intersection can be written as following:

$$\tilde{\lambda}_{\mathrm{C}}(\boldsymbol{n}_{\mathrm{C}}) := \begin{cases} \tilde{\lambda}_{\mathrm{C}}(\boldsymbol{n}_{\mathrm{C}}) \text{ if } \hat{\lambda}(\phi(\boldsymbol{n}_{\mathrm{C}})) \leq \lambda(\phi(\boldsymbol{n}_{\mathrm{C}})) \\ \rho(\boldsymbol{n}_{\mathrm{C}}) \quad \text{if } \hat{\lambda}(\phi(\boldsymbol{n}_{\mathrm{C}})) > \lambda(\phi(\boldsymbol{n}_{\mathrm{C}})) \end{cases} . \tag{12}$$

By applying this cutting step to every single ray, ${}^{t}\tilde{K}$ represents the outer hull ${}^{t}A_{\mathrm{M}}$, afterwards.

5.2 Model Based 2D Contour Computation

In the following, the method is shortly summarized to create the silhouette of the 3D ray model in a given pose $(\boldsymbol{R}_{\mathrm{MC}}, \boldsymbol{t}_{\mathrm{MC}})$. The resulting silhouette shall be represented by $K = (\boldsymbol{m}, \lambda)$. The reference point $\boldsymbol{m}$ is set to $\mathcal{P}(\tilde{\boldsymbol{m}}_{\mathrm{C}})$. It remains to determine the values $\lambda(\phi)$.

In the continuous case, for each value ϕ, a set of $B(\phi) \subset \tilde{W}$ of 3D rays can be found, defined by following relation: $\boldsymbol{n}_{\mathrm{C}} \in B \Leftrightarrow \phi(\boldsymbol{n}_{\mathrm{C}}) = \phi$, where the function ϕ is defined equal Section 5.1. For each direction $\boldsymbol{n}_{\mathrm{C}}$ also the length $\hat{\lambda}(\phi(\boldsymbol{n}_{\mathrm{C}}))$ can be determined. The value $\lambda(\phi)$ is now given by the following equation:

$$\lambda(\phi) = \max\{\hat{\lambda}(\phi(\boldsymbol{n}_{\mathrm{C}})) \,|\, \boldsymbol{n}_{\mathrm{C}} \in B(\phi)\}. \tag{13}$$

In the discrete case, the set $B(\phi)$ is given by the relation $\boldsymbol{n}_{\mathrm{C}} \in B \Leftrightarrow \phi - \Delta\phi \leq \phi(\boldsymbol{n}_{\mathrm{C}}) < \phi + \Delta\phi$, where $\Delta\phi$ denotes the maximum 3D angle between two neighboring 3D rays. The value $\lambda(\phi)$ is then defined equal to the continuous case.

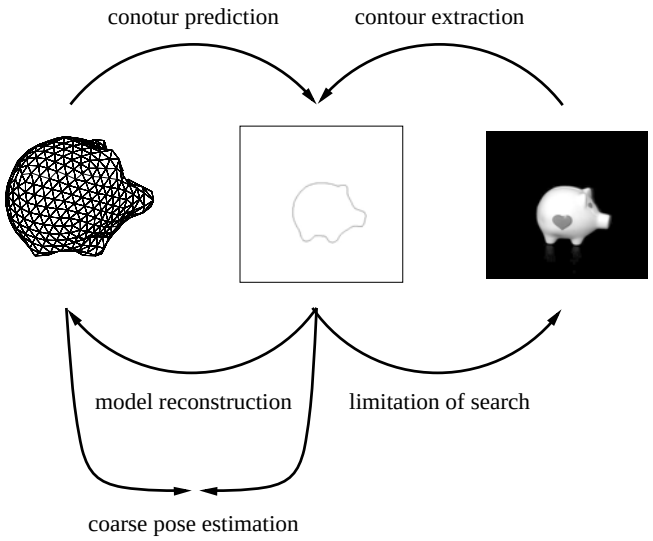


Figure8. Data flow between 2D and 3D ray representations. A set of extracted 2D contours is used for 3D model generation. The model can then be used to predict a 2D contour for a given pose of the object.

5.3 System Integration

Now the 2D and 3D ray representations are tied together for contour based object tracking. In Figure 8 the connections between both representations and the system integration is clarified. The important aspect for object tracking is the possibility, that for each angle ϕ we get an interval $I(\phi) = [\lambda_s(\phi), \lambda_e(\phi)]$, which is used to limit the contour point search in the 2D image plane (compare Section 2). Thus equation (5) becomes

$$\lambda^*(\phi) = \underset{\lambda(\phi)\in I(\phi)}{\operatorname{argmin}} \int_0^{2\pi} \left[\frac{\alpha(\phi)\left|\frac{d}{d\phi}\lambda(\phi)\right|^2 + \beta(\phi)\left|\frac{d^2}{d\phi^2}\lambda(\phi)\right|^2}{2} - \left|\frac{d}{d\lambda}\varrho_{\boldsymbol{m}}(\phi,\lambda)\right|^2 \right] d\phi \quad (14)$$

Having both, the 2D contour and the 3D model, a coarse 3D pose estimation is possible, which is needed to perform a 3D motion estimation and prediction with a Kalman filter. For this, we have to define a distance function $\operatorname{dist}(\boldsymbol{v}, \boldsymbol{v}')$, which measures the similarity of two 2D contours $\boldsymbol{v}$ and $\boldsymbol{v}'$. Then the 3D pose, defined by $\boldsymbol{R}^*_{\mathrm{MC}}$ and $\boldsymbol{t}^*_{\mathrm{MC}}$, can be estimated by

$$\left(\boldsymbol{R}^*_{\mathrm{MC}}, \boldsymbol{t}^*_{\mathrm{MC}}\right)^T = \underset{\boldsymbol{R}_{\mathrm{MC}}, \boldsymbol{t}_{\mathrm{MC}}}{\operatorname{argmin}} \operatorname{dist}(\boldsymbol{v}, \boldsymbol{v}') \quad (15)$$

where $\boldsymbol{v}$ corresponds to the extracted 2D contour and $\boldsymbol{v}'$ is the 2D contour of the model under the transformation $\boldsymbol{R}_{\mathrm{MC}}$ and $\boldsymbol{t}_{\mathrm{MC}}$.

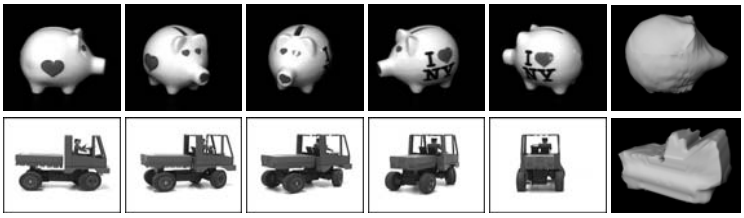


Figure9. Some images out of the sequence taken from [15] and one recorded by us, and reconstructed coarse models.

6 Experimental Evaluation

We have conducted several experiments, concerning the 3D reconstruction, 2D prediction and tracking. In Figure 9 some images of two sequences are shown with the models reconstructed models. The first sequence totally consits of 36, the second of 9 images. Several other objects (for example, see Figure 10, right) haven been reconstructed. We have tested convex objects as well as concave objects. Although the reconstructed 3D shape of concave objects does not look quite well, the model is accurate enough concerning the accuracy of the predicted 2D contour.

The computation time for model generation depending on the 2D and 3D ray resolution are summarized in Table 1. For the models in Figure 9 and 8 an amount of 960 3D rays and 36 2D rays have been chosen, which means, that also model generation can be done in real–time. This is important for an online model building during tracking in the future. Another example for model generation can be seen in Figure 10 showing a toy train, following an elliptic way. Although the real pose of the object was estimated very roughly by deviding the whole rotation angle by the number of images and assuming a vertical rotation axis, the reconstructed model nearly has the same side relations as the original one: $1 : 1.45 : 1.95$ (model) and $1 : 1.37 : 1.63$ (original object).

In Figure 10 three predicted contours and the corresponding extracted contours for the model from Figure 9 are shown, which proves the accuracy of contour prediction. The computation time for 2D contour prediction can be seen in Table 1. The mean error in the ray length is about 2% to 5% of the extracted ray length, varying in the different views.

Finally, in Figure 11 a result for tracking a moving toy train is shown. The extracted motion path is overlayed the image.

7 Discussion and Future Work

In this paper we have presented an efficient combination of a 2D contour representation and a 3D modelling technique. The motivation of the work has been to provide a mechanism for 2D contour prediction of rigid objects moving in 3D. This is an important aspect for contour based tracking algorithms in the case of natural scenes with heterogeneous background.

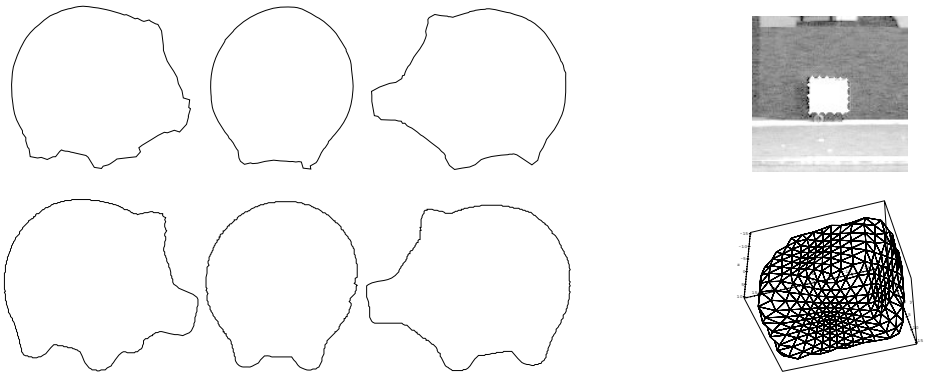


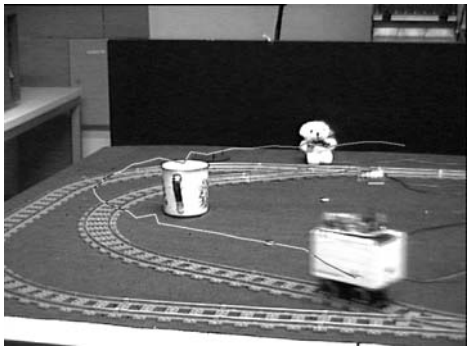
Figure10. Left: comparison between predicted 2D contour (first row) and extracted 2D contour (second row). The prediction is accurate enough to successfully limit the search space for 2D rays. Right: one image out of a sequence following a toy train (top) and the reconstructed coarse model (bottom).

	model generation			contour generation		
	number of 3D rays			number of 3D rays		
number of 2D rays	60	960	15360	60	960	15360
18	1	8	145	1	8	149
36	1	8	143	1	10	179
180	1	7	119	5	26	423
360	2	8	120	10	51	733

Table1. Computation time for processing one image during the process of model and contour generation, depending on the 2D and 3D ray resolution (in msec. with an SGI Onyx R10000)

A short introduction to a radial representation of 2D contours and an energy description for data driven contour extraction has been followed by a similar representation of 3D objects. The 3D model of an object can be built from a set of different 2D contours. Well known methods from shape from contour have been applied. The key idea has been, to make use of the similar representations, namely a radial representation for the 2D as well as for the 3D case, to achieve real–time performance. This has been proven in the experimental part.

It is worth noting, that we did not want to implement an accurate 3D model generation algorithm. In the context of active contours, a coarse 2D contour initialization is sufficient to extract the object's contour in the following energy minimization step. Thus, the 3D visual hull of the object is sufficient for modelling the contour changes during tracking. At present, we are integrating this approach in our real–time object tracking system. Preliminary results show, that the performance of the system can be increased for partial



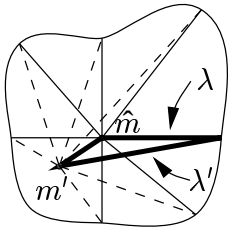


Figure11. Left: tracking result (3D path of the moving object overlayed the image) in the case of partial occlusions. Right: resampling of a 2D contour with a different reference point.

occlusions of the moving object. Also, the 3D pose estimation over time can be used for qualitative 3D motion estimation (for example, the object is moving toward the camera) of the moving object, as it has been reported in [7].

In the near future we focus on the following. First, during 3D model generation, the 3D rays, which have been cut due to a segmentation error, cannot increase in length. This results in holes on the objects surface. Secondly, the initial 3D pose estimation of the object is a time consuming task, in the case that no a priori pose information is possible. Then, the complete parameter space $\boldsymbol{R}_{\text{MC}}$ and $\boldsymbol{t}_{\text{MC}}$ has to be searched. After the initialization, this can be done in real–time by assuming, that the pose of the object is changing slowly.

Appendix: Technical Details for the Contour Comparison

In our implementation of the system, we made the assumption of orthogonal projection. Then we are able to calculate the 2D translation $\boldsymbol{t}_{\text{2D}}$ between the extracted and the model contour: $\boldsymbol{t}_{\text{2D}} = \boldsymbol{b} - \hat{\boldsymbol{b}}$, where $\boldsymbol{b}$ denotes the balance point of the extracted contour and $\hat{\boldsymbol{b}}$ of the projection of the model silhouette.

To compare the two contours, we have to superimpose them and also their reference points must be equal. We set the projection of the reference point of the model to 0_2, so we only translate the extracted contour by subtracting the translation vector $\boldsymbol{t}_{\text{2D}}$ from the reference point $\boldsymbol{m}$. Then we resample the extracted contour with the new reference point 0_2 by trigonometric calculations illustrated in Figure 11.

Afterwards for each new defined ray of the 2D representation, the corresponding ray length of the model projection can be determined by equation 13. To get the difference measure in equation (15) between the two contours, we integrate over the square difference of the accoring ray lengths.

References

1. M. Berger. Tracking rigid and non polyhedral objects in an image sequence. In *Scandinavian Conference on Image Analysis*, pages 945–952, Tromso (Norway), 1993.
2. R. Beß. Objekterkennung mit dem Extended Gaussian Image. Technical report, Diploma thesis, Lehrstuhl für Mustererkennung (Informatik 5), Universität Erlangen–Nürnberg, Erlangen, 1993.
3. 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.
4. T.F. Cootes, C.J. Taylor, D.H. Cooper, and J. Graham. Active shape models — their training and applications. *Computer Vision Graphics and Image Processing*, 61(1):38–59, 1995.
5. R. Curwen and A. Blake. Dynamic contours: Real–time active splines. In A. Blake and A. Yuille, editors, *Active Vision*, pages 39–58. MIT Press, Cambridge, 1992.
6. J. Denzler. *Active Vision for Real–Time Object Tracking*. Dissertation, Technische Fakultät, Universität Erlangen–Nürnberg, Erlangen, in preparation, 1997.
7. J. Denzler and H. Niemann. 3d data driven prediction for active contour models based on geometric bounding volumes. *Pattern Recogniton Letters*, 17(11):1171–1178, 1996.
8. J. Denzler and H. Niemann. Real–time pedestrian tracking in natural scenes. In G. Sommer, K. Daniliidis, and J. Pauli, editors, *Computer Analysis of Images and Patterns, CAIP'97, Kiel 1997*, pages 42–49, Berlin, 1997. Lecture Notes in Computer Science.
9. D.B. Gennery. Visual tracking of known 3d objects. *International Journal of Computer Vision*, 7(3):243–270, 1992.
10. C. Harris. Tracking with rigid models. In A. Blake and A. Yuille, editors, *Active Vision*, pages 59–74. MIT Press, Cambridge, 1992.
11. M. Kass, A. Wittkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 2(3):321–331, 1988.
12. K.F. Lai and R.T. Chin. Deformable contours: Modelling and extraction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(1):1–20, 1996.
13. A. Laurentini. How far 3d shapes can be understood from 2d silhouettes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(2):188–195, 1995.
14. 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.
15. S. A. Nene, S. K. Nayar, and H. Murase. Columbia object image library (coil-20), 1996. http://www.cs.columbia.edu/CAVE/coil-20.html.
16. R. Ronfard. Region-based strategies for active contour models. *International Journal of Computer Vision*, 13(2):229–251, 1994.
17. C.G. Small. Reconstructing convex bodies from random projected images. *The Canadian Journal of Statistics*, 19(4):341–347, 1991.
18. S.M. Smith and J.M. Brady. Asset-2: Real–time motion segmentation and shape tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):814–820, 1995.
19. D. Terzopoulos and R. Szeliski. Tracking with Kalman snakes. In A. Blake and A. Yuille, editors, *Active Vision*, pages 3–20. MIT Press, Cambridge, 1992.