Learning, Tracking and Recognition of 3D Objects

J. Denzler, R. Beß[1], J. Hornegger, H. Niemann, D. Paulus

The following was printed in the proceedings
IROS 94
München
Sept. 1994

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

Learning, Tracking and Recognition of 3D Objects

J. Denzler, R. Beß, J. Hornegger, H. Niemann, D. Paulus
Lehrstuhl für Mustererkennung (Informatik 5)
Universität Erlangen–Nürnberg
Martensstr. 3, D–91058 Erlangen, Germany

Abstract

In this contribution we describe steps towards the implementation of an active robot vision system. In a sequence of images taken by a camera mounted on the hand of a robot, we detect, track, and estimate the position and orientation (pose) of a three–dimensional moving object. The extraction of the region of interest is done automatically by a motion tracking step. For learning 3–D objects using two–dimensional views and estimating the object's pose, a uniform statistical method is presented which is based on the Expectation–Maximization–Algorithm (EM–Algorithm). An explicit matching between features of several views is not necessary. The acquisition of the training sequence required for the statistical learning process needs the correlation between the image of an object and its pose; this is performed automatically by the robot. The robot's camera parameters are determined by a hand/eye-calibration and a subsequent computation of the camera position using the robot position. During the motion estimation stage the moving object is computed using active, elastic contours (snakes). We introduce a new approach for online initializing the snake on the first images of the given sequence, and show that the method of snakes is suited for real time motion tracking.

1 Introduction

Current research in image analysis increasingly focuses on applications involving moving parts using real world scenes. Neither the objects nor the visual devices are stationary. Active manipulation of the cameras as well as the objects in the scene are common to the systems. The well known Marr paradigm [11] is extended to embody this situation. In addition to an *interpretation* of the visual data, *actions* have to be performed. A closed loop of active devices and the interpretation of the image analysis system imposes new constraints on the architecture. For such a system to operate, various interacting modules have to contribute their results to a general control module. A list of references for known algorithms for tracking and classifying moving objects can be found in [8].

In this paper we present a system design for image analysis in a robot application. The so called *problem domain* consists of a moving toy train [2] on a table; [3] a stereo camera system and a third camera mounted on a robot's arm observe this scene. Here, we investigate the problem of following a moving object in this scene and estimating its pose. Furthermore, the system includes the capability of learning new objects automatically using different views of an object. First, an offline calibration of the robot's camera is done (section 2). Using the calibration data different views of the object are captured with known camera parameters. A statistical training method is applied for learning the object (section 3). It renders the computation of 3–D structure without feature matching of several two–dimensional views.

After the training stage, the object starts moving in the scene. An attention module has to detect and track the moving object in the subsequent images (section 4). Tracking is performed by active contour models [7]. Unlike other work we describe an online initialization of the snake — a closed contour around the moving object, see section 4 — on the object's contour in the first image. The tracking task with a modified snake algorithm satisfies the requirements with respect to real time computation and tracking performance. While tracking one object we compute point features of the object itself. These features can now be given to the classification stage for pose estimation. The algorithm for the calculation of the position of the object is also derived from the EM–Algorithm. The paper concludes with experimental results revealing advantages and problems of the approach (section 5), and an outlook for future research.

2 Camera Calibration

The training algorithm in the statistical approach for object recognition, described in the following section, needs a sample of images. This sample has to show different 2–D views of an object taken from randomly chosen camera positions. An approach to get these images is to use the camera which is mounted on the gripper of the robot and determine its position from the information the robot can give about the position of its gripper. If the transformation between the position of the camera and the robot's gripper is known, we will compute for each view both the camera position and the corresponding robot position. After that we can stop at the computed

[2] Lego toy train: donation of LEGO, Germany
[3] No domain dependend knowledge is used in this model

position and capture an image. Hereby image acquisition can be performed automatically.

We use the approach of Tsai and Lenz, described in detail in [16] to determin the transformation between gripper and camera. The determination is trivial, if the position and orientation of an object in the robot coordinate system are given, and if the relative position and orientation of the camera with respect to the object is observable. However, an accurate determination of an object's position with respect to the robot coordinate system is a nontrivial problem; Tsai and Lenz note that some authors treat this problem as part of a large non-linear optimization process. The approach of Tsai and Lenz allows to measure the position and orientation of gripper and camera in two different coordinate systems. From a sequence of at least three pairs of positions they compute the transformation. The camera position will be determined by *camera calibration*: From the image of a pattern with a number of accurately measured calibration points the parameters of the transformation between world coordinates (defined with respect to the calibration pattern) and camera coordinates will be computed. The transformation between robot coordinate system and the gripper is measured by the robot's hardware.

The camera model chosen for the mapping between object and image is the pin hole model with radial lens distortion. If from at least 11 points both the world and the image coordinates are known, one may compute the parameters of the mapping, including the transformation between camera and world coordinates [9].

The transformations explained in this paper are described using the Denavit–Hartenberg notation [4]. Every transformation of a coordinate system A in a coordinate system B is defined by a homogeneous transformation ${}^{B}\boldsymbol{H}_{A}$. With $\boldsymbol{h}_A = (x, y, z, 1)_t$ defining a point $(x, y, z)_t$ in coordinate system A the following equations hold:

$${}^{B}\boldsymbol{H}_{A}\boldsymbol{h}_A = {}^{A}\boldsymbol{H}_{B}{}^{-1}\boldsymbol{h}_A \quad \text{and} \quad {}^{C}\boldsymbol{H}_{B}\,{}^{B}\boldsymbol{H}_{A}\boldsymbol{h}_A = {}^{C}\boldsymbol{H}_{A}\boldsymbol{h}_A$$

Figure 1 graphically shows the homogeneous transformations and the coordinate systems used in this paper: G_i and G_k the gripper coordinate system G at position i and position k; C_i and C_k the camera coordinate system at position i and position k; the robot coordinate system R and the world coordinate system W.

From a sequence of images $\boldsymbol{f}_k, k = 1, ..., n$ taken from different positions one may determine the transformations ${}^{C_k}\boldsymbol{H}_{W}$ by camera calibration and the transformations ${}^{R}\boldsymbol{H}_{G_k}$ from the position measured by the robot. The rotation matrix ${}^{G}\boldsymbol{R}_{C}$ and the translation vector ${}^{G}\boldsymbol{t}_{C}$ which together define the gripper/camera transformation ${}^{G}\boldsymbol{H}_{C}$ will be determined separately.

Any rotation may be defined by a rotation matrix $\boldsymbol{R}$ or one–to–one by a single vector $\boldsymbol{p}_R$, which is given by $\boldsymbol{p}_R := 2\sin\frac{\theta_R}{2}\boldsymbol{r}_R \;\; (0 \le \theta_R \le \pi)$, where $\boldsymbol{r}_R = (n_1\ n_2\ n_3)_t$ is the rotation axis with direction cosines n_1, n_2, n_3 and θ_R is the rotation angle. Because a rotation axis does not change during a rotation around itself it may be computed as the Eigenvector of $\boldsymbol{R}$ corresponding to the Eigenvalue 1, which is the solution of the equation $\boldsymbol{R}\boldsymbol{r}_R = \boldsymbol{r}_R$.

Vice versa $\boldsymbol{R}$ may be computed from $\boldsymbol{p}_R$ by:

$$\boldsymbol{R} = \left(1 - \frac{|\boldsymbol{p}_R|^2}{2}\right)\boldsymbol{I} + \frac{1}{2}(\boldsymbol{p}_R\boldsymbol{p}_{R\ t} + \alpha\boldsymbol{S}(\boldsymbol{p}_R)), \qquad (1)$$

where $\alpha = \sqrt{4 - |\boldsymbol{p}_R|^2}$ and $\boldsymbol{S}$ stands for a 3×3 skew matrix:

$$\boldsymbol{S}(\boldsymbol{p}) := \begin{pmatrix} 0 & -p_z & p_y \\ p_z & 0 & -p_x \\ -p_y & p_x & 0 \end{pmatrix}.$$

For details and a derivation of equation 1 see [16].

For better readability the following abbreviations for the rotation axis and the rotation angle of a rotation matrix ${}^{B}\boldsymbol{R}_{A}$ will be used:

$${}^{B}\boldsymbol{r}_A := \boldsymbol{r}_{B\boldsymbol{R}_A} \qquad {}^{B}\theta_A := \theta_{B\boldsymbol{R}_A}$$

If we denote the transformations between two camera or gripper positions by ${}^{C_k}\boldsymbol{H}_{C_i} = {}^{C_k}\boldsymbol{H}_{W}\ {}^{W}\boldsymbol{H}_{C_i}$ and ${}^{G_k}\boldsymbol{H}_{G_i} = {}^{G_k}\boldsymbol{H}_{R}\ {}^{R}\boldsymbol{H}_{G_i}$, the computation of the transformations will be performed in the following two steps:

1. At first the rotation axis ${}^{G}\boldsymbol{r}_{C}$ and the rotation angle ${}^{G}\theta_{C}$ are computed. Together they define the rotation matrix ${}^{G}\boldsymbol{R}_{C}$.
   The auxiliary vector ${}^{G}\boldsymbol{p}'_{C}$ is defined by:
   $${}^{G}\boldsymbol{p}'_{C} := \frac{1}{2\cos({}^{G}\theta_{C}/2)}{}^{G}\boldsymbol{p}_{C} = \frac{1}{\sqrt{4 - |{}^{G}\boldsymbol{p}_{C}|^2}}{}^{G}\boldsymbol{p}_{C}\ .$$
   For each pair of stations i, k one gets a system of equations linear in the components of the vector ${}^{G}\boldsymbol{p}'_{C}$:
   $$\boldsymbol{S}({}^{G_k}\boldsymbol{p}_{G_i} + {}^{C_k}\boldsymbol{p}_{C_i}){}^{G}\boldsymbol{p}'_{C} = {}^{C_k}\boldsymbol{p}_{C_i} - {}^{G_k}\boldsymbol{p}_{G_i}\ . \qquad (2)$$
   Because $\boldsymbol{S}(\boldsymbol{v})$ is singular for all $\boldsymbol{v}$ one consequently needs at least two pairs of stations to get an unambiguous solution by the minimization of the mean square error.
   Using ${}^{G}\boldsymbol{p}'_{C}$ the values for ${}^{G}\theta_{C}$ will be determined by
   $${}^{G}\theta_{C} = 2\arctan|{}^{G}\boldsymbol{p}'_{C}| \quad \text{and} \quad {}^{G}\boldsymbol{p}_{C} = \frac{2\,{}^{G}\boldsymbol{p}'_{C}}{\sqrt{1 + |{}^{G}\boldsymbol{p}'_{C}|^2}}\ .$$

2. If ${}^{G}\boldsymbol{R}_{C}$ is known, ${}^{G}\boldsymbol{t}_{C}$ will be determined.
   Two pairs of stations i, k result in two sets of three linear equations:
   $$({}^{G_k}\boldsymbol{R}_{G_i} - I){}^{G}\boldsymbol{t}_{C} = {}^{G}\boldsymbol{R}_{C}\ {}^{C_k}\boldsymbol{t}_{C_i} - {}^{G_k}\boldsymbol{t}_{G_i}\ . \qquad (3)$$
   Again the solution is determined by the minimization of the mean square error.

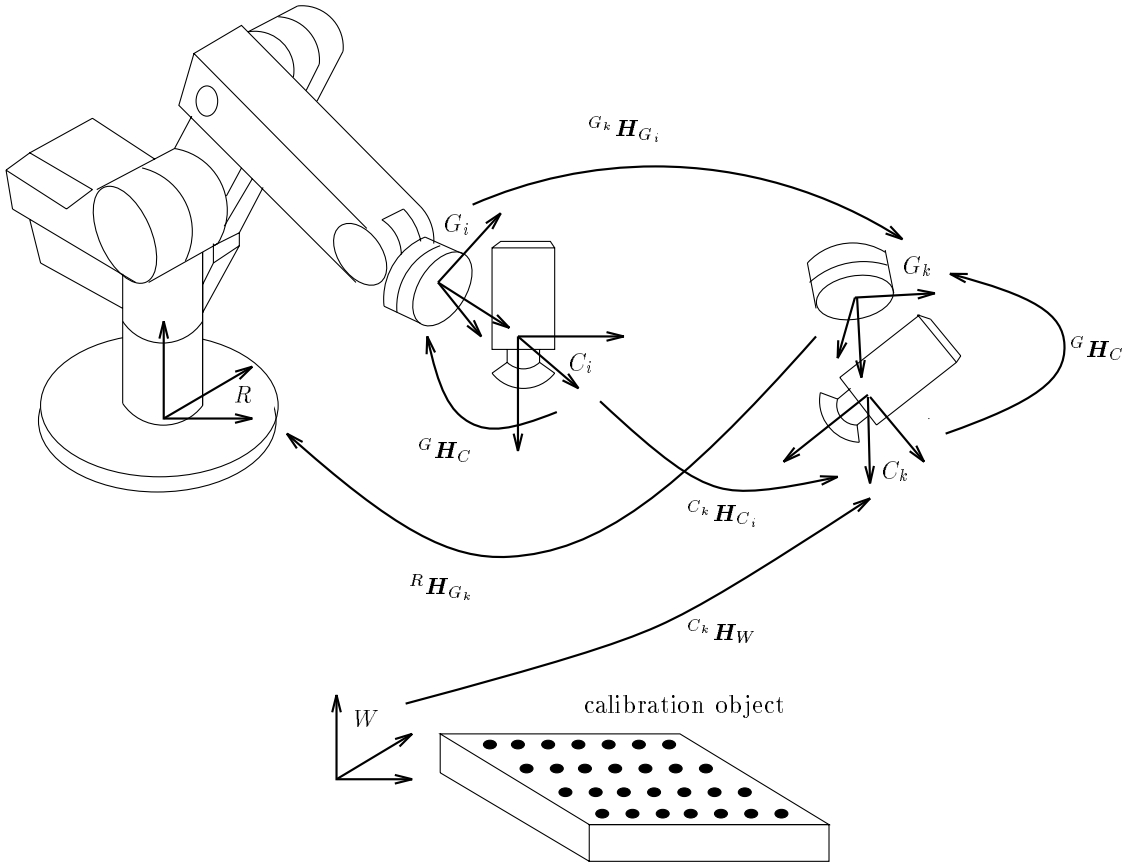

Figure 1: Coordinate systems and transformations as explained in section 2. The constant transformation $^{G}\boldsymbol{H}_{C}$ is computed from at least three different positions of camera and gripper, two of which – C_i, G_i and C_k, G_k – are sketched in the figure.

Proofs for the equations 2 and 3 can be found in [16].

The time needed for the calculation of the gripper/camera transformation is dominated by the time needed for feature extraction during camera calibration.

3 Learning and Classification

Using the calibrated camera we can generate a set of 2–D views of an object including the parameters for rotation and translation in the 3–D space. Based on these training data the following section describes an approach for the learning and recognition of objects in the given environment. Motivated by segmentation errors and the instability of edge features we choose a statistical framework with parameterized probability density functions. Each observable feature is understood as a random variable and associated with a density function for modeling its uncertainty. Since the distribution of features occurring in the image plane depends on the object's pose, we presume that these density functions result from projections of the feature distributions in the model space. The density function for rotated and translated model features projected into the image plane can be computed using the algebra of random variables (see [15]). The problem of learning objects in this mathematical context corresponds to the parameter estimation of the mixture distribution in the model space. The training samples are transformed model features, where the transformation is a non injective mapping, and additionally we do not know the matching between the image features in the learning views. The classification is based on a combination of maximization of a posteriori probabilities and a parameter estimation problem for pose determination. We leave out an abstract mathematical description of the training and classification algorithms (see [6]) and explain a special statistical model which was implemented and evaluated in a number of experiments. Nevertheless, we need some notational remarks. The set

of observable features in the j–th sample scene is denoted by $\boldsymbol{O}_j = \{\boldsymbol{O}_{j,1}, \boldsymbol{O}_{j,2}, \ldots, \boldsymbol{O}_{j,m}\}$; let the number of views for training be J. Furthermore, we abbreviate the set of model features with $\boldsymbol{C}_\kappa = \{\boldsymbol{C}_{\kappa,1}, \boldsymbol{C}_{\kappa,2}, \ldots, \boldsymbol{C}_{\kappa,n}\}$. For example, the features used here are points. Since a scene includes also background features we introduce $\boldsymbol{C}_{\kappa,0}$, where all observable features of the background are assigned to. Each feature $\boldsymbol{C}_{\kappa,i}$ $(i \geq 1)$ of a three–dimensional model is assumed to be normally distributed. Additionally, the statistical behavior of the background features $\boldsymbol{C}_{\kappa,0}$ among individual instances of observable scenes is described by uniform distributions. In [17] it was shown by statistical tests that point features in grey–level images satisfy these assumptions. We use these results and generalize for 3–D model primitives that this observation is based on a Gaussian distribution of features in the model space. If an affine mapping, given by a matrix $\boldsymbol{R}$ and translation $\boldsymbol{t}$, transforms a normally distributed variable with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{K}$, then the result is again Gaussian distributed. The mean vector and the covariance matrix of the transformed random variable are $\boldsymbol{R\mu} + \boldsymbol{t}$ and $\boldsymbol{RKR}^T$ (see [15]).

In the implemented algorithms an arbitrary model $\boldsymbol{C}_\kappa$ is represented by a set of 3–D points. A linear transform which projects a model point $\boldsymbol{C}_{\kappa,i}$ to its corresponding image point $\boldsymbol{O}_{j,k}$ describes the geometric relation between model points and the set of scene features $\boldsymbol{O}_j$.

$$\boldsymbol{O}_{j,k} \quad = \quad \boldsymbol{RC}_{\kappa,i} + \boldsymbol{t}. \tag{4}$$

The set of observable points is postulated to be normally distributed, that means the location of the model points and its projections are assumed to be Gaussian random vectors.

The training phase now proceeds as follows: We take images of the 3–D object from different views and detect automatically edges, vertices, and corners. The rotation and translation of the object is equivalent to the movement of the robot's arm with its camera. For each image in the training sample set the rotation and translation parameters are known due to the fact that the camera parameters are given. Unknown information during the learning phase is on the one hand the three–dimensional structure of the object, i.e. the normal distributions of 3–D points, and on the other hand the matching between the image points of several views. This should be learned during the off–line training step.

Under the idealized assumption that the model features are pairwise independent and characterized by a mixture density function the a priori density functions for an observed scene $\boldsymbol{O}_j$ and its pose can be written as

$$p(\boldsymbol{O}_j, \boldsymbol{R}_j, \boldsymbol{t}_j | \boldsymbol{B}) = \prod_{k=1}^{m} \sum_{i=0}^{n} p(\boldsymbol{C}_{\kappa,i})\, p(\boldsymbol{O}_{j,k}, \boldsymbol{R}_j, \boldsymbol{t}_j | \boldsymbol{a}_{\kappa,i}), \tag{5}$$

where $\boldsymbol{B} = \{\boldsymbol{a}_{\kappa,1}, \boldsymbol{a}_{\kappa,2}, \ldots, \boldsymbol{a}_{\kappa,n}\}$ represents the set of parameters of the model's density function, i.e. the set of covariance matrices and mean vectors, and $p(\boldsymbol{C}_{\kappa,i})$ the probability of observing the model feature $\boldsymbol{C}_{\kappa,i}$ or a projection of it. We know that all observable 2–D point features are normally distributed, but the matching among model and image features is missing. Therefore, it is proposed that each observable feature can be generated by projections of each model feature with a special probability. An explicit matching between model and scene primitives can be avoided and the probability of observing a scene feature $\boldsymbol{O}_{j,k}$ is consequently

$$p(\boldsymbol{O}_{j,k}, \boldsymbol{R}_j, \boldsymbol{t}_j | \boldsymbol{B}) = \sum_{i=0}^{n} p(\boldsymbol{C}_{\kappa,i}) p(\boldsymbol{O}_{j,k}, \boldsymbol{R}_j, \boldsymbol{t}_j | \boldsymbol{a}_{\kappa,i}). \tag{6}$$

Since the mean $\boldsymbol{\mu}_i$, covariance matrix $\boldsymbol{K}_i$, and weight $p(\boldsymbol{C}_{\kappa,i})$ for each model feature $\boldsymbol{C}_{\kappa,i}$ are unknown, they have to be estimated from the learning samples. The estimation of parameters and probabilities of mixture densities is most widely done by maximum–likelihood estimates and the algorithms are well studied for non transformed random variables (see [12]). Due to the fact that both the origin of each observable image feature and the 3–D structure of the object are unknown, it is suggested to use in the given situation the iterative **E***xpectation* **M***aximization Algorithm* developed by Dempster, Laird, and Rubin [3], which is suitable for obtaining maximum–likelihood estimates from incomplete data. The purpose of the EM–Algorithm is the computation of the density of the three–dimensional model features using only the observable 2–D features and the information about the object's pose of each view. It is necessary to emphasize that the projection of model features to the image plane has no inverse. The application of the EM–Algorithm to our training problem results in the following training formulas

$$\hat{p}(\boldsymbol{C}_{\kappa,l}) = \frac{1}{J\,m} \sum_{j=1}^{J} \sum_{k=1}^{m} p(\boldsymbol{C}_{\kappa,l} \mid \boldsymbol{O}_{j,k}, \boldsymbol{R}_j, \boldsymbol{t}_j, \boldsymbol{a}_{\kappa,l}). \tag{7}$$

for the probabilities of each feature and the formula for mean estimates

$$\hat{\boldsymbol{\mu}}_i = \left(\sum_{j=1}^{J} \sum_{k=1}^{m} p(\boldsymbol{C}_{\kappa,i} \mid \boldsymbol{O}_{j,k}, \boldsymbol{a}_{\kappa,i}) \boldsymbol{R}_j^T (\boldsymbol{R}_j \boldsymbol{K}_i \boldsymbol{R}_j^T)^{-1} \boldsymbol{R}_j \right)^{-1}$$
$$\sum_{j=1}^{J} \sum_{k=1}^{m} p(\boldsymbol{C}_{\kappa,i} | \boldsymbol{O}_{j,k}, \boldsymbol{a}_{\kappa,i}) \boldsymbol{R}_j^T (\boldsymbol{R}_j \boldsymbol{K}_i \boldsymbol{R}_j^T)^{-1} (\boldsymbol{O}_{j,k} - \boldsymbol{t}_j). \tag{8}$$

The covariance matrices $\mathbf{K}_i$ can be iteratively computed solving the system of equations

$$\sum_{j=1}^{J} \sum_{k=1}^{m} p(\boldsymbol{C}_{\kappa,i} | \boldsymbol{O}_{j,k}, \boldsymbol{a}_{\kappa,i}) \boldsymbol{R}_j^T \boldsymbol{D}_{i,j}^{-1} \hat{\boldsymbol{D}}_{i,j} \boldsymbol{D}_{i,j}^{-1} \boldsymbol{R}_j \;=$$

$$\sum_{j=1}^{J}\sum_{k=1}^{m} p(\boldsymbol{C}_{\kappa,i}|\boldsymbol{O}_{j,k},\boldsymbol{a}_{\kappa,i})\boldsymbol{R}_j^T\boldsymbol{D}_{i,j}^{-1}\boldsymbol{S}_{j,k}\boldsymbol{S}_{j,k}^T\boldsymbol{D}_{i,j}^{-1}\boldsymbol{R}_j, \qquad (9)$$

where $\hat{\boldsymbol{D}}_{i,j} = \boldsymbol{R}_j\hat{\boldsymbol{K}}_i\boldsymbol{R}_j^T$, $\boldsymbol{D}_{i,j} = \boldsymbol{R}_j\boldsymbol{K}_i\boldsymbol{R}_j^T$, and $\boldsymbol{S}_{j,k} = \boldsymbol{O}_{j,k} - \boldsymbol{R}_j\boldsymbol{\mu}_i - \boldsymbol{t}_j$. Formulas (7) and (8) can be implemented easily. The system of equations in (9) has to be solved. If the dimensions of model and of its projected point features are known, the equations can be solved a priori symbolically. For the same reason, even the matrix inversions in (8) can be computed symbolically and explicitly implemented. Finally, the formulas show that the complexity of learning is bounded by $O(Jmn)$. The classification problem in our system should include both the identification and the pose determination of the moving object. The computation of rotation and translation can be done by optimizing the a posteriori density function $p(\boldsymbol{R}_j, \boldsymbol{t}_j \mid \boldsymbol{O}_j)$ with respect to $\boldsymbol{R}_j$ and $\boldsymbol{t}_j$. The object decision is made by maximizing the a posteriori probabilities of all object classes. Since the matching among model and image features is unknown, we have again a set of incomplete training samples for estimating these parameters. The EM–theory yields the following optimization problem for computing the pose parameters:

$$(\hat{\mathbf{R}}_j, \hat{\mathbf{t}}_j) = \underset{(\hat{\mathbf{R}}_j,\hat{\mathbf{t}}_j)}{\operatorname{argmax}} \sum_{k=1}^{m}\sum_{l=0}^{n} \frac{p(\boldsymbol{O}_{j,k}, \boldsymbol{C}_{\kappa,l}|\boldsymbol{R}_j, \boldsymbol{t}_j, \boldsymbol{a}_{\kappa,l})}{p(\boldsymbol{O}_{j,k}|\boldsymbol{R}_j, \boldsymbol{t}_j, \boldsymbol{B})} \log p(\boldsymbol{C}_{\kappa,l})p(\boldsymbol{O}_{j,k} \mid \hat{\boldsymbol{R}}_j, \hat{\boldsymbol{t}}_j, \boldsymbol{a}_{\kappa,l}). \qquad (10)$$

Different techniques for solving this problem can be used [14]. Continuous local optimization algorithms using gradient information cannot be used without reserve, since they are only suitable for detecting local extrema. If the search space is discrete, we can use combinatorial optimization algorithms like simulated annealing, which guarantee to find a global maximum.

4 Motion

In the previous section we have presented a method for learning and classification of 3–D objects. Obviously the success and computation time of pose estimation is considerably influenced by the number of background features. This section is devoted to the problem of reducing background features, i.e. we will describe an algorithm for the automatic extraction of moving objects from a sequence of images in order to put these objects to the learning and classification stage of our experimental environment. For motion tracking we use active contours; first we summarize the principles of active contours.

The energy minimizing model of active contours (*snakes*) was first introduced by [7]. 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 $\mathcal{S}$ can be defined as a parametric function $\boldsymbol{u}(l)$

$$\boldsymbol{u}(l) = (x(l), y(l)),\ l \in [0, n-1] \qquad (11)$$

with

$$x(l) \in [0, X_{max}], \quad y(l) \in [0, Y_{max}] \qquad (12)$$

where X_{max} and Y_{max} are usually given by the size of the input image. Such an active contour has an energy $\boldsymbol{E}$ defined by

$$\boldsymbol{E} = \sum_{i=0}^{n-1} E_{int}(\boldsymbol{u}(i)) + E_{ext}(\boldsymbol{u}(i)). \qquad (13)$$

The external forces can be the image $\boldsymbol{f}(x,y)$ or the edge strength of an image, for example $E_{ext}(\boldsymbol{u}(i)) = -|\nabla \boldsymbol{f}(\boldsymbol{u}(i))|^2$. In the case of the edge strength as the external force during the energy minimization the snake will be pushed to strong edges, for example the contour of an object. Further information concerning the snake model and its behavior can be found in [7] and [10]. In several papers, for example [2], [10], the advantages of snakes for object tracking are shown. Given an image sequence $\boldsymbol{f}_1(x,y), \boldsymbol{f}_2(x,y), \ldots, \boldsymbol{f}_n(x,y)$ including just 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 $\boldsymbol{u}_t(l)$ of image $\boldsymbol{f}_t(x,y)$ on the image $\boldsymbol{f}_{t+1}(x,y)$. If the object is moving sufficiently slow, the snake will extract the object's contour in the image $\boldsymbol{f}_{t+1}(x,y)$ by energy minimization.

Due to lack of space, this is only a short summary of the snake procedure and the ideas of tracking moving objects using snakes. The main problem described in this chapter is the initialization of the snake by the use of the first image. Most researchers use an interactive initialization on the first image [7], [10]. Here we will describe a simple method for automatic initializition of the snake on the second image of a sequence of images, containing one moving object. For the initialization we assume a non–moving camera. Since all investigations are aimed to real time tracking, speed is an important aspect to consider.

For real time motion detection and tracking we need a fast, computationally efficient attention module for the detection of a moving object in a scene. Our first version of the attention module uses a simple difference images algorithm based on low resolution images. Given a sequence of images $\boldsymbol{f}_1(x,y), \boldsymbol{f}_2(x,y), \ldots, \boldsymbol{f}_n(x,y)$ of image size $640 * 480$, we proceed in the following way (start the algorithm at $t = 0$):

1. Resample the images $\boldsymbol{f}_t(x,y)$ and $\boldsymbol{f}_{t+1}(x,y)$ to an image size of $128 * 128$.

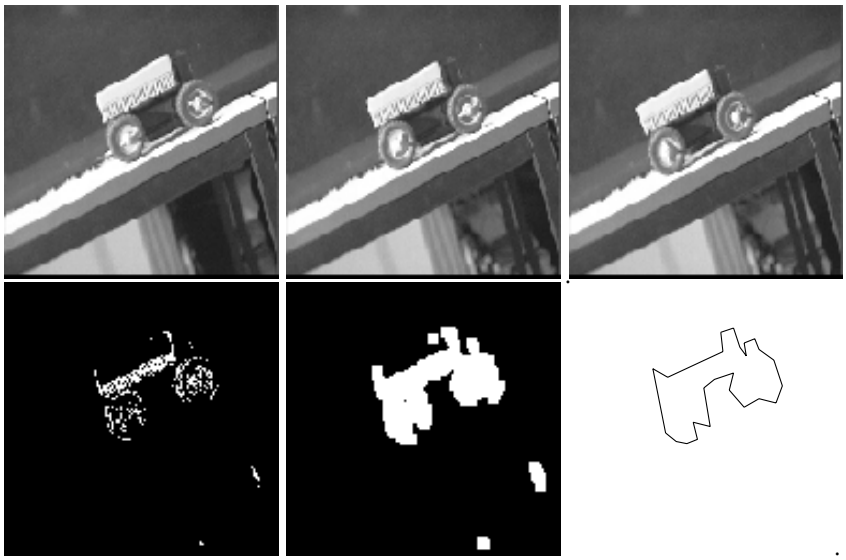

Figure 2: From top left to lower right: Image 0, 10, 20, of an image sequence of 30 images; difference image and smoothed difference image (region of interest); straight line approximation as initialization of snake. The toy train is moving downhill from left to right.

2. Compute the difference image $\boldsymbol{D}_{t+1}(x,y) = \boldsymbol{f}_t(x,y) - \boldsymbol{f}_{t+1}(x,y)$, where

$$\boldsymbol{D}_{t+1}(x,y) = \begin{cases} 0 & ; \quad |\boldsymbol{f}_t(x,y) - \boldsymbol{f}_{t+1}(x,y)| < \delta \\ 1 & ; \quad \text{otherwise} \end{cases}$$

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

4. If there is no significant region, we assume that there is no moving object. Take the next image and go to step 1.

5. Extract a chain code of the boundary of the binary region of interest. Approximate the chain code with straight line segments.

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).

7. Use the start points (or the end points) of the straight line segments as the initial snake elements positions.

8. Start the snake algorithm. The result is the contour of the moving object.

The steps 1 - 3 build the attention module. This method is simple and not a general way to detect and track motion, but experiments show that it is sufficient in this problem domain, due to the fact that an active contour collapses to one point if no external forces are in the image [10]. So a coarse initialization round the moving object is sufficient. One fast way to achieve this coarse initialization is the described difference images algorithm. The method fails, if the difference image does not completely cover the object, or in the presence of strong background edges near the object.

Now the tracking algorithm can start with the active contour. As a result one gets the contour of the moving object and we can now extract the moving object itself [1] to classify it using the algorithms described in the previous section.

5 Experiments and Results

In this section we present first experiments and results of our system. The software is written using the object–oriented image analysis system ἵππος described in [13] on HP 735 workstations. The computation of the transformation between gripper and camera needs 11 `msec` when

24 pairs of positions are used, the robot takes 48 `sec` to approach these positions. Segmentation of the image taken at each position and camera calibration may be done during robot movement.

For the estimation of means, covariances, and weights a parameter initialization of the density function for each feature is required. The number of features and initial estimates of means, covariance matrices, and weights have to be established. Presently we use in our experiments views where no occlusion occurs. For simple polyedric objects the method produces satisfactory results, if we determine the number of features using one view. The mean vectors are initialized by the observable 2–D point features, with the depth value set to zero. Empirically, 40–50 views are sufficient for learning an object with 15 characteristic point–features. Although the convergence rate of the EM–Algorithm was expected to be considerably low (see [3]),

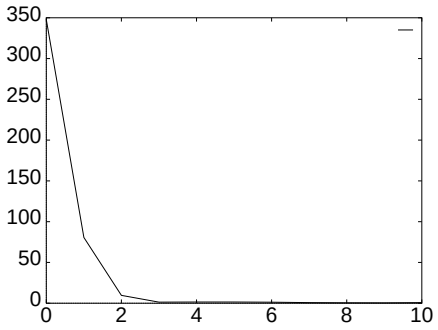

Figure 3: The convergence of learning.

the learning process converged in average after 10 iterations. The change in the mean–square error of one randomly chosen estimated mean out of 15 object features with an increasing number of iterations is shown in Figure 3. The time needed for one iteration using a very general C++ implementation of the learning formula (8) takes 97.98 seconds with 50 training views. The memory requirements are constant for each iteration. The experiences with methods for pose estimation show that gradient techniques are very sensitive to the initialization of rotation and translation parameters. The reason for this observation is that the sample data are limited to the observed features; furthermore the density function for the a posteriori probability $p(\boldsymbol{R}, \boldsymbol{t} \mid \boldsymbol{O}_j)$ is a multimodal function (see Figure 4 for two degrees of freedom) and gradient methods do not guarantee to detect the global maximum.

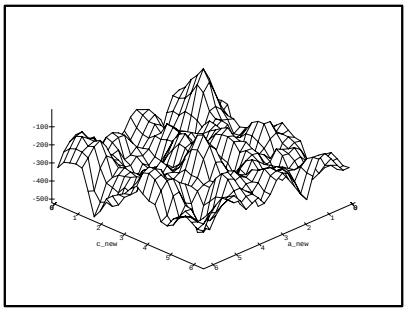

Figure 4: Multimodal density function

A detailed description of the motion detection, the tracking algorithm, the modification to the original snake algorithm, and implementation details can be found in [5].

# total	# correct	# false	% correct
65	43	22	66.15

Figure 5: Results of automatic initialization.

In Figure 5 the results of the initialization experiments are shown. After initializing the snake on the contour of the moving object this object is correctly tracked over an average number of 90 images. Three example images out of our image sequence are shown in Figure 2. Also the result of the difference image between image No. 0 and No. 1 and the straight line approximation of the chain code of the contour of the region of interest can be seen. In Figure 6 the result of the tracking with the snake algorithm is shown. Computing the contour of the moving object needs 8.81 seconds for the 30 images. Because of a frame rate of 8.5 images per second this means a real time factor of 2.5 in our prototypical implementation.

6 Conclusion

In this paper we have described a three–stage system for an active robot system application. A moving object in a scene has been tracked with a robot's camera and its pose can be estimated. The camera calibration – needed in the learning and classification module – and the mathematics for the computation of the camera position from the robot position were introduced. We have then presented a new learning and classification module for 3–D objects based on the EM–Algorithm. To extract a moving object, whose pose has to be computed out of a sequence of images, an active vision method with snakes is used. Therefore we have presented a new method for automatic initialization of the snake on the contour of the object in the first image. The computation time of a prototypical implementation of the snake algorithm raises our hope for a real time motion tracking and classification in the near future, by the use of a workstation cluster with up to eight HP-735 workstations.

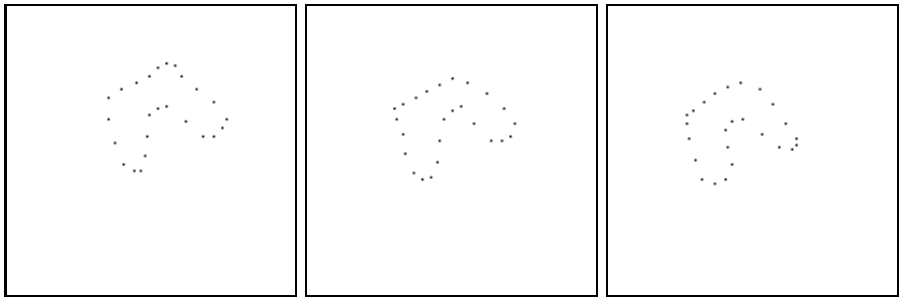

Figure 6: Results of tracking the toy train with snakes (image 0, 10, 20).

References

1. S.M. Ali and R.E. Burge. A new algorithm for extracting the interior of bounded regions based on chain coding. *Computer Vision, Graphics, and Image Processing*, 43(3):256–264, 1988.
2. M. Berger. Tracking rigid and non polyedral objects in an image sequence. In *Scandinavian Conference on Image Analysis*, pages 945–952, Tromso (Norway), 1993.
3. A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society, Series B (Methodological)*, 39(1):1–38, 1977.
4. J Denavit and R. S. Hartenberg. A kinematic notation for lower–pair mechanisms based on matrices. *Journal of Applied Mechanics*, 77:215–221, 1955.
5. J. Denzler. Aktive Konturen. Technical Report MEMO/IMMD 5/BV/AV1/93, Lehrstuhl für Mustererkennung (Informatik 5), Universität Erlangen, 1993.
6. J. Hornegger. A Bayesian approach to learn and classify 3–d objects from intensity images. In *Proc 12th International Conference on Pattern Recognition*, Jerusalem, Israel, to appear October 1994. IEEE Computer Society Press.
7. M. Kass, A. Wittkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 2(3):321–331, 1988.
8. D. Koller. *Detektion, Verfolgung und Klassifikation bewegter Objekte in monokularen Bildfolgen am Beispiel von Straßenverkehrsszenen*, volume 13 of *Dissertationen zur künstlichen Intelligenz*. infix, St. Augustin, 1992.
9. R. Lenz. Linsenfehlerkorrigierte Eichung von Halbleiterkameras mit Standardobjektiven für hochgenaue 3D-Messungen in Echtzeit. In E. Paulus, editor, *Proceedings 9. DAGM-Symposium*, Informatik Fachberichte 149, pages 212–216, Berlin, 1987. Springer.
10. 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.
11. D. Marr. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. W.H. Freeman and Company, San Francisco, 1982.
12. H. Niemann. *Klassifikation von Mustern*. Springer, Berlin, 1983.
13. D. W. R. Paulus. *Objektorientierte und wissensbasierte Bildverarbeitung*. Vieweg, Braunschweig, 1992.
14. W.H. Press, B.P. Flannery, S. Teukolsky, and W.T. Vetterling. Numerical recipes - the art of numerical computing, c version. Technical report, 35465-X, 1988.
15. M. D. Springer. *The Algebra of Random Variables*. Wiley Publications in Statistics. John Wiley & Sons, Inc., New York, 1979.
16. R. Y. Tsai and Lenz. R. Real time versatile robotics hand/eye calibration using 3d machine vision. In *International Conference on Robotics and Automation*, pages 554–561, Philadelphia, Pa., Apr. 1988. IEEE.
17. W. M. Wells III. *Statistical Object Recognition*. PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Massachusetts, February 1993.