An Efficient and Robust Real-Time Contour Tracking System

Giorgio Panin Alexander Ladikos Alois Knoll

Technical University of Munich
Chair for Robotics and Embedded Systems
Boltzmannstrasse 3, 85748 Munich, Germany
{panin,ladikos,knoll}@in.tum.de

Abstract

In this paper we present an efficient and robust real-time system for object contour tracking in image sequences. The developed application partly relies on an optimized implementation of a state-of-the-art curve fitting algorithm, and integrates important additional features in order to achieve robustness while keeping the speed of the main estimation algorithm. An application program has been developed, which requires only a few standard libraries available on most platforms, and runs at video frame rate on a common PC with standard hardware equipment.

1 Motivation and Scope of the Present Work

The general problem of object contour tracking in image sequences is an important and challenging topic in the computer vision community; as many researchers already pointed out, an advanced contour tracking technique can provide crucial information for many image understanding problems and, at the same time, allows the development of efficient and useful working applications in many fields of interest. We refer the reader to [1] for a survey of these applications and the related references.

Among the currently available methodologies, a very appealing one is the *Contracting Curve Density* (CCD) algorithm: this method has been recently developed and presented in [3] as a state-of-the-art improvement over other advanced techniques such as the Condensation algorithm [6], and it has been shown to overperform them in many different estimation tasks. Nevertheless, its higher complexity has been initially considered as an obstacle to an effective real-time implementation, even in the simplified form named Real-Time CCD from the same authors [4], and a working online version still had to be investigated.

Moreover, in order to realize an autonomous and robust tracking system, some important additional issues have to be taken into account; one of them is the possibility of automatic initialization and reinitialization of the system, both at the beginning of the tracking, and in case of tracking loss. Nevertheless, one also expects that a "well behaving" tracking system does not need a global estimation procedure during most of the tracking task, or, in the ideal case, only at the beginning. The global initialization module is computationally more demanding than the online tracker, and a frequent reinitialization would significantly slow down the performance of the system.

Therefore, most of the efforts in realizing such a system have to be focused on a robust real-time performance of the contour tracker itself, in terms of speed, accuracy and *convergence area* for the parameter search.

All of the above mentioned issues, and other relevant aspects, constitute the main motivation of the present work; it will be shown how it contributes to the state-of-the-art in contour tracking systems with the following achievements:

- The real-time CCD algorithm [4] has been reimplemented with a significant number of critical *speedups* with respect to the originally proposed version.
- A global initialization and reinitialization module has been developed and introduced into the system
- A trajectory coherence test module based on simple kinematic models gives the possibility of reliably checking for a tracking loss condition at any time

As a result of these improvements, we were able to build a working real-time application, that can easily run on a common PC equipped with standard video hardware at video frame rate, while preserving the robustness and precision of the base algorithm.

The paper is organized as follows: in Section 2, a general overview of the tracking system will be given; Section 3 will then focus on the core CCD tracking algorithm, and the most important implementation speedups will be described;

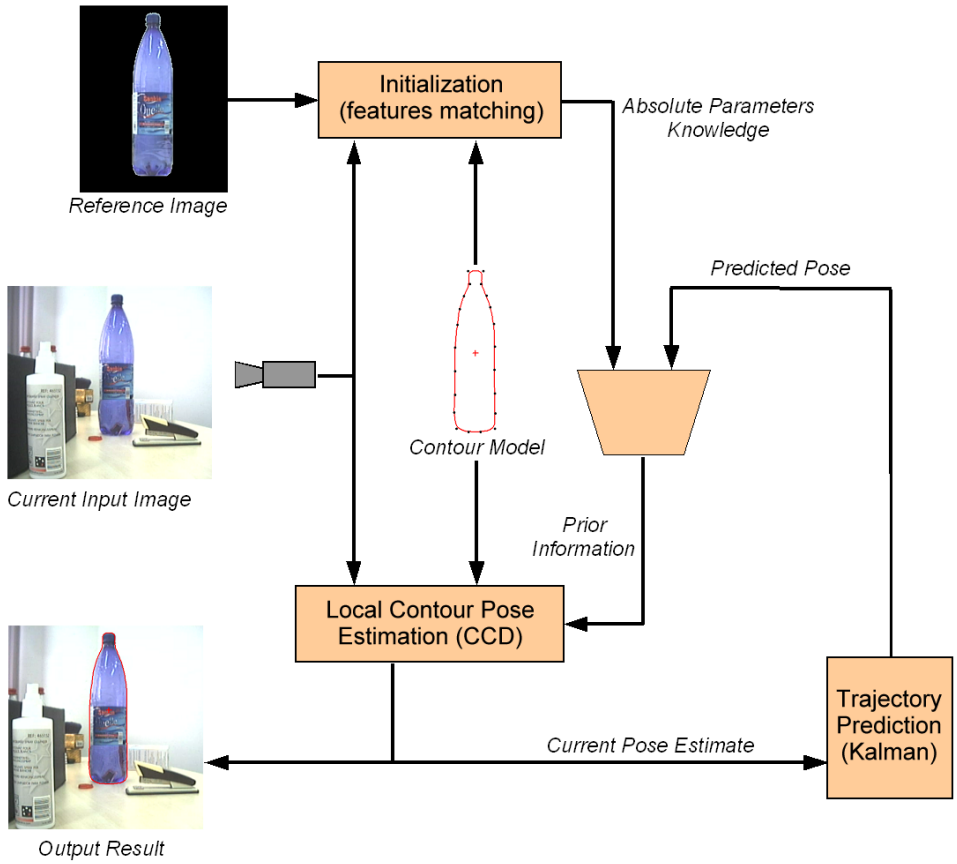

Figure 1. Overall system architecture

Section 4 describes the global initialization search module, and the trajectory checking module; experimental results and conclusions are finally given in Section 5.

2 System Overview

A general overview of the system is given here, particularly focusing on the main purpose and interactions of the different modules involved, and the information flow between them. Referring to Fig. 1, we can see that the system consists of several interacting modules, exchanging online and offline data. A description of the system behavior is given by the following abstract procedure:

Setup: In the first step, the user needs to specify a basic *model description* for the contour tracking task; this operation needs to be done only once for a given object to be tracked. The base contour shape of the object is taken by simply placing a few *control points* of the curve on the image, in order to satisfactory match the visible object contour. The result of this process is the base *contour shape model* and the *reference image*, that are stored in files; both will be used in the subsequent tracking steps.

Initial Search: When activating the contour tracking engine, the system searches for the object in the incoming image stream, by using a robust *feature matching* system against the reference image, in order to initialize and reinitialize the tracking algorithm. The obtained pose parameters and their uncertainty constitute the *prior* (or *absolute*) knowledge of the system state, in absence of reliable predictions from the tracker.

Local Estimation: The CCD module holds the contour model of the system, and receives the online RGB image stream coming from the video camera. It refines the contour estimation coming from the prior knowledge, and gives as output the current estimate of the contour pose parameters. The estimated pose parameters are used afterwards for two main purposes: to make a pose *prediction* for the next timestep, and to check the *trajectory coherence* in order to reinitialize the system in case of object loss.

Prediction and Trajectory Check: The trajectory prediction and check is accomplished by a simple but suitable

kinematic model, via a steady-state *Kalman Filter*. The filter uses the current estimated pose from the CCD algorithm as state measurement. If the normalized measurement residuals at the given timestep do not exceed a given treshold, the predicted pose will be accepted as prior knowledge for the next frame CCD estimation. Otherwise, the feature matching module will be called in order to reinitialize the system.

3 The Improved Real-Time CCD Algorithm

This Section considers in some detail the main speed-up improvements over the original CCD algorithm [3][4], starting from a hypothetical initial estimate with a given uncertainty.

In order to describe the algorithm improvements, we recall its main underlying principles here.

3.1 Contour Model Parametrization

As already mentioned in Section 2, the object contour model is defined as a parametric model, that describes the shape of the object in terms of a base shape and a limited number of degrees of freedom (*dof*), chosen in order to specify the allowed displacements and deformations of the base curve.

In this implementation we first consider planar rigid objects with a single contour, moving in 3D space, and we model the contour as a continuous, differentiable and open *quadratic B-Spline* in $\Re^2$. The contour representation is given by a set of N_C *control points* lying on the *xy* plane, $Q = \{\mathbf{q}_0, \mathbf{q}_1, ..., \mathbf{q}_{N_C-1}\}$ and a piecewise quadratic *Basis Function* $B(s)$, and the curve equation is given by

$$\mathbf{c}(s) = \sum_{n=0}^{N_C-1} \mathbf{q}_n B(s-n) \tag{1}$$

In the present work splines-models consisting of approximately 20 control points were used. In order to search for the best matching contour pose, the contour points are then transformed in 3D space according to a *6-dof* parametrized mapping, that can be regarded as the *state-space* of the contour.

The overall mapping consists therefore of a rigid roto-translation, followed by perspective projection of the given contour points on the image screen; for this purpose, it is first necessary to specify in advance the internal camera acquisition model, given by a set of *intrinsic* parameters $\mathbf{C}$ that are obtained off-line via a common calibration procedure. The space transformation for the object contour will be hereafter denoted by a vector $\mathbf{\Phi}$, so that the overall transformation from a space point $\mathbf{x}$ to a screen point $\mathbf{q}$ is given by

$$\mathbf{q} = T(\mathbf{x}, \mathbf{\Phi}, \mathbf{C}) \tag{2}$$

In particular, we decided to represent the rigid rotations by means of a common *yaw-pitch-roll* parametrization; in order to avoid singularities, the current frame rotation parameters are referred to the *previous* estimated frame attitude, instead of the fixed camera frame. After each successful estimation, the current rotation matrix is updated by multiplying it with the estimated one, and the result is re-orthogonalized in order to avoid numerical drifts.

Therefore, our transformation vector will be

$$\mathbf{\Phi} \equiv [\alpha, \beta, \gamma, t_x, t_y, t_z]; \quad \mathbf{\Phi} \in \Re^6 \tag{3}$$

with three rotation angles about the respective axes of the reference frame, and three translations t_x, t_y, t_z.

3.2 Step 1 of CCD: Learn Local Statistics

Once the model parameter space has been specified, the optimization algorithm is initialized by setting the global estimated parameter mean and covariance matrix $(\mathbf{m}_{\mathbf{\Phi}}, \mathbf{\Sigma}_{\mathbf{\Phi}})$ to the prior knowledge $(\mathbf{m}^*_{\mathbf{\Phi}}, \mathbf{\Sigma}^*_{\mathbf{\Phi}})$, where the statistics are modeled by multivariate Gaussian distributions; in order to speed up the computations, all the covariance matrices have been chosen to be *diagonal*, that is to neglect the statistical interaction between the parameter components. This assumption has no significant influence on the optimization behavior, while dramatically increasing the computation speed.

A precomputed set of K *sample points* $\left\{\mathbf{c}_1^{(0)}, ..., \mathbf{c}_K^{(0)}\right\}$, which are uniformly distributed w.r.t. the contour parameter *s*, is then transformed according to the current hypothesis $\mathbf{\Phi} = \mathbf{m}_{\mathbf{\Phi}}$, by applying the spline transformation; the parameter K is also specified in advance. The result is a set of local image positions $\{\mathbf{c}_1, ..., \mathbf{c}_K\}$ and *normal* vectors $\{\mathbf{n}_1, ..., \mathbf{n}_K\}$, from where to start collecting the local statistics.

The next step consists of taking a set of points along the normal directions on both sides of the curve for each sample position. For this purpose, first the planar contour normals, lying on the *xy* plane, are computed offline. We decided to set a *fixed distance* h along the line segments on which the set of points are collected and evaluated according to the hypothetical uncertainty of the current evaluation step, by following the heuristic rule

$$h^2 = \left[\det\left(\mathbf{\Sigma}_{\mathbf{\Phi}}\right)\right]^{1/N} \tag{4}$$

with $N = 6$ the parameter space dimension. This idea, which differs from the original CCD formulation, allows a significant computational saving by giving a constant shape for the local statistics *weighting function* at each sample

point, thus avoiding many expensive nonlinear function evaluations, as will be explained later. For a contour that encloses a limited area, moreover, some attention has been paid to limit the search distance on the internal side, in order to avoid "crossing" the opposite boundary, thus sampling pixels from the wrong area.

In order to get the actual sample segments on the screen, the normal segments are then uniformly sampled along their length, and the resulting body-frame points are stored.

After the search distance h has been set, we proceed by assigning two suitable weighting functions w_1, w_2 to the pixels $\mathbf{v}_{kl}$ along the normal for the two sides of the curve, by dividing the normal into an overall number of L equally spaced sample points ($L/2$ for each contour side). Following the suggested rules in [3], we decided to define these functions as

$$w_{1/2}(d_l) := C\left(\frac{a_{1/2}(d_l)-\gamma_1}{1-\gamma_1}\right)^6\left[e^{-d_l^2/2\hat{\sigma}^2}-e^{-\gamma_2}\right]^+ \tag{5}$$

where $d_l = d(\mathbf{v}_{kl}, \mathbf{c}_k)$ is the distance to the curve (with sign) along the normal, C a normalization factor, and

$$a_1(d) := \frac{1}{2}\left[erf\left(\frac{d}{\sqrt{2}\sigma}\right)+1\right]; \quad a_2 := 1-a_1 \tag{6}$$

the smooth *assignment* functions to the two sides of the curve, with $\gamma_1 = 0.5$ (disjoint weight assignment) and $\gamma_2 = 4$ for the truncated Gaussian in (5). Moreover, the standard deviation $\hat{\sigma}$ has been chosen such as to cover the specified distance h. That means

$$\hat{\sigma} = \max\left[\frac{h}{\sqrt{2\gamma_2}}, \gamma_4\right]; \quad \sigma = \frac{1}{\gamma_3}\hat{\sigma} \tag{7}$$

with the two additional constants $\gamma_3 = 6$ (linear dependence between σ and $\hat{\sigma}$) and $\gamma_4 = 4$ (minimum weighting window width). The L distances d_l and the assignment and weighting function values are then computed in advance and stored in arrays.

During the current optimization step, each body sample point is transformed according to the current pose parameters, and the $2K$ local unnormalized RGB statistics up to the second order are collected as specified in the original CCD algorithm (see [3] and [4])

$$\begin{aligned}\mathbf{m}_{ks}^{(0)} &= \sum_{l=1}^{L} w_{kls} \\ \mathbf{m}_{ks}^{(1)} &= \sum_{l=1}^{L} w_{kls}\mathbf{I}_{kl} \\ \mathbf{m}_{ks}^{(2)} &= \sum_{l=1}^{L} w_{kls}\mathbf{I}_{kl}\mathbf{I}_{kl}^T\end{aligned} \tag{8}$$

with $s = 1, 2$ (for each curve side), $k = 1, ..., K$, and $\mathbf{I}_{kl}$ the observed pixel colors at the sample point locations.

In order to perform the *blurring* step along the contour, as recommended in [3] with exponential filtering, we managed to simplify the operation by using a *constant* set of filtering coefficients for the whole contour, independent from the current shape $\mathbf{\Phi}$; every contour point $k = 1, ..., K$ will receive a contribution from each other point k_1 given by a coefficient

$$b(k-k_1) = (\lambda/2)\exp(-\lambda|k-k_1|) \tag{9}$$

with $\lambda = 0.4$ in our implementation.

The blurred statistics are then mixed with their time-smoothed version, that has been filtered with a simple first-order exponential decay rule, and the resulting quantities are normalized, thus giving a set of expected color values and covariance matrices for each sample position k and each side of the contour.

The complete normalized local statistics set will be indicated with $S = \left(\bar{\mathbf{I}}_k^{(1)}, \bar{\mathbf{I}}_k^{(2)}, \mathbf{\Sigma}_k^{(1)}, \mathbf{\Sigma}_k^{(2)}\right)$ with $2K$ color mean vectors $\bar{\mathbf{I}}_k^{(1)}, \bar{\mathbf{I}}_k^{(2)}$ and (3×3) positive definite covariance matrices $\mathbf{\Sigma}_k^{(1)}, \mathbf{\Sigma}_k^{(2)}$, for each side of the curve and each sample position.

3.3 Step 2 of CCD: Compute the Cost Function and its Derivatives

Here the collected local statistics are used in order to obtain the matching function, its gradient and the Hessian matrix, which will be used to update the pose parameters.

By following [3], we write the general cost function formulation, with a slightly different notation:

$$E = E_1(\mathbf{\Phi}, \mathbf{m}_{\mathbf{\Phi}}^*, \mathbf{\Sigma}_{\mathbf{\Phi}}^*) + E_2(\mathbf{\Phi}, S) \tag{10}$$

where

$$E_1 := \frac{1}{2}(\mathbf{\Phi}-\mathbf{m}_{\mathbf{\Phi}}^*)^T(\mathbf{\Sigma}_{\mathbf{\Phi}}^*)^{-1}(\mathbf{\Phi}-\mathbf{m}_{\mathbf{\Phi}}^*) + C(\mathbf{\Sigma}_{\mathbf{\Phi}}^*) \tag{11}$$

is the prior log-likelihood according to a Gaussian distribution, and

$$\begin{aligned}E_2 &:= -\log\prod_{k,l} p(\mathbf{I}_{kl}|\mathbf{\Phi}, S) = \\ &= \sum_{k,l}\left[\tfrac{1}{2}\left(\mathbf{I}_{kl}-\hat{\mathbf{I}}_{kl}\right)^T\hat{\mathbf{\Sigma}}_{kl}^{-1}\left(\mathbf{I}_{kl}-\hat{\mathbf{I}}_{kl}\right) + C\left(\hat{\mathbf{\Sigma}}_{kl}\right)\right]\end{aligned} \tag{12}$$

with normalization factor $C\left(\hat{\mathbf{\Sigma}}_{kl}\right)$, is the log-probability of the observed colors $\mathbf{I}_{kl}$ w.r.t. the pixel statistics $\left(\hat{\mathbf{I}}_{kl}, \hat{\mathbf{\Sigma}}_{kl}\right)$, with $\hat{\mathbf{I}}_{kl}$ given by

$$\hat{\mathbf{I}}_{kl} = a_1(d_l)\,\bar{\mathbf{I}}_k^{(1)} + (1-a_1(d_l))\,\bar{\mathbf{I}}_k^{(2)} \tag{13}$$

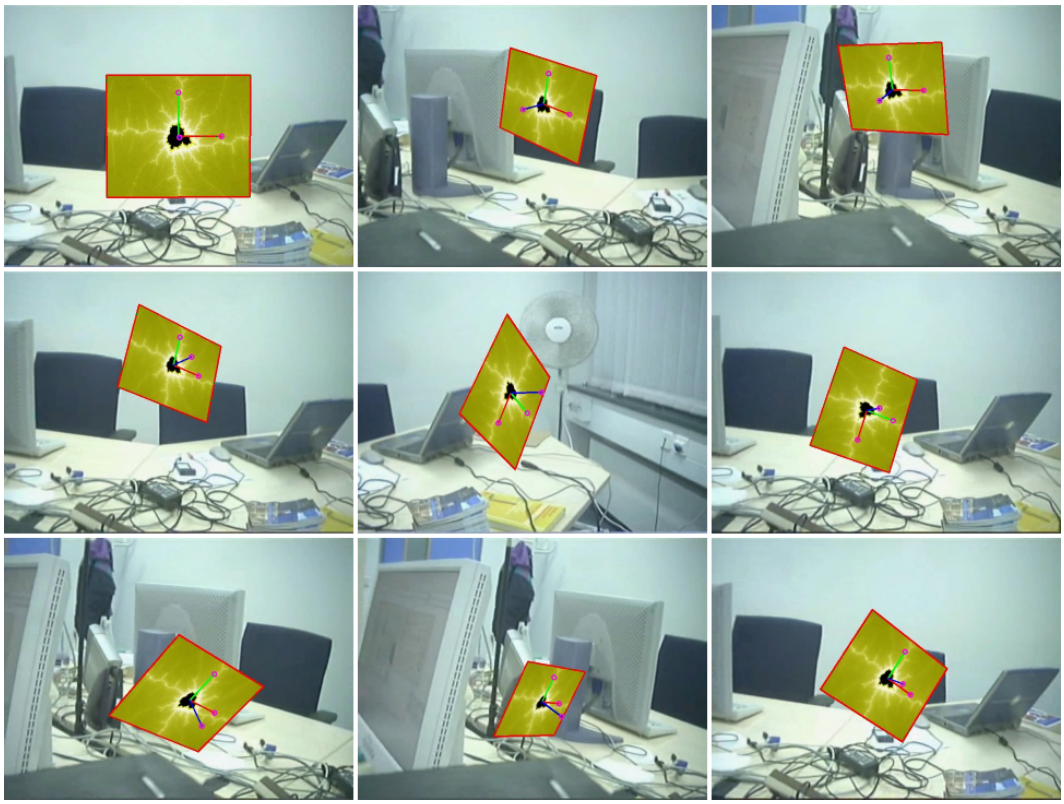

Figure 2. Results of the tracking on a simulated scenario

Regarding the local covariance matrices $\hat{\mathbf{\Sigma}}_{kl}$ in (12), we decided not to follow the rule (13), as proposed in [3], because this leads to a high computational cost for the derivatives of E_2. Instead, we decided to assign the covariance matrices with the "hard" (not differentiable) rule

$$\hat{\mathbf{\Sigma}}_{kl} = \frac{1}{2}\left[\left(1 + sign\left(d_l\right)\right)\mathbf{\Sigma}_k^{(1)} + \left(1 - sign\left(d_l\right)\right)\mathbf{\Sigma}_k^{(2)}\right] \tag{14}$$

which becomes a good approximation of the smooth assignment a_1 when $\sigma \to 0$.

The rule (14) is perhaps the most crucial speedup of this implementation, since it allows to reformulate the cost function E_2 in a standard *weighted nonlinear least squares* form

$$E_2 = \frac{1}{2}\sum_{k,l}\left(\mathbf{I}_{kl} - \hat{\mathbf{I}}_{kl}\left(\mathbf{\Phi}\right)\right)^T \hat{\mathbf{\Sigma}}_{kl}^{-1}\left(\mathbf{I}_{kl} - \hat{\mathbf{I}}_{kl}\left(\mathbf{\Phi}\right)\right) \tag{15}$$

where the only dependence on $\mathbf{\Phi}$ has been put into evidence, and the first term of the sum has therefore been neglected for the optimization.

For this problem, the Gauss-Newton approximation to the Hessian matrix can be adopted. First, the gradient is computed as

$$\nabla_{\mathbf{\Phi}} E_2 = -\sum_{k,l} J_{a_1}^T \hat{\mathbf{\Sigma}}_{kl}^{-1}\left(\mathbf{I}_{kl} - \hat{\mathbf{I}}_{kl}\right) \tag{16}$$

with

$$J_{a_1}\left(\mathbf{\Phi}, \mathbf{v}_{kl}\right) := \left(\bar{\mathbf{I}}_k^{(1)} - \bar{\mathbf{I}}_k^{(2)}\right)\left(\nabla_{\mathbf{\Phi}} a_1\left(d_l\right)\right)^T \tag{17}$$

Afterwards, the Gauss-Newton approximation to the Hessian matrix is given by

$$H_{\mathbf{\Phi}} E_2 = \sum_{k,l} J_{a_1}^T \hat{\mathbf{\Sigma}}_{kl}^{-1} J_{a_1} \tag{18}$$

The overall gradient and Hessian matrices for the optimization are obtained by adding the prior cost function derivatives, and the Newton optimization step can finally be performed as

$$\begin{aligned} \mathbf{m}_{\mathbf{\Phi}} &\to \mathbf{m}_{\mathbf{\Phi}} - \left(H_{\mathbf{\Phi}} E\right)^{-1} \nabla_{\mathbf{\Phi}} E \\ \mathbf{\Sigma}_{\mathbf{\Phi}} &\to \alpha \mathbf{\Sigma}_{\mathbf{\Phi}}; \qquad \alpha < 1 \end{aligned} \tag{19}$$

where the covariance matrix is updated as well by an exponential decay rule.

A *confirmation* check before every parameter update, as recommended in [3], is also employed, with an improved speed due to the use of diagonal covariance matrices. The optimization then restarts from step 1 with the updated parameters, and repeats the whole procedure until convergence; usually 10 optimization steps give satisfactory results.

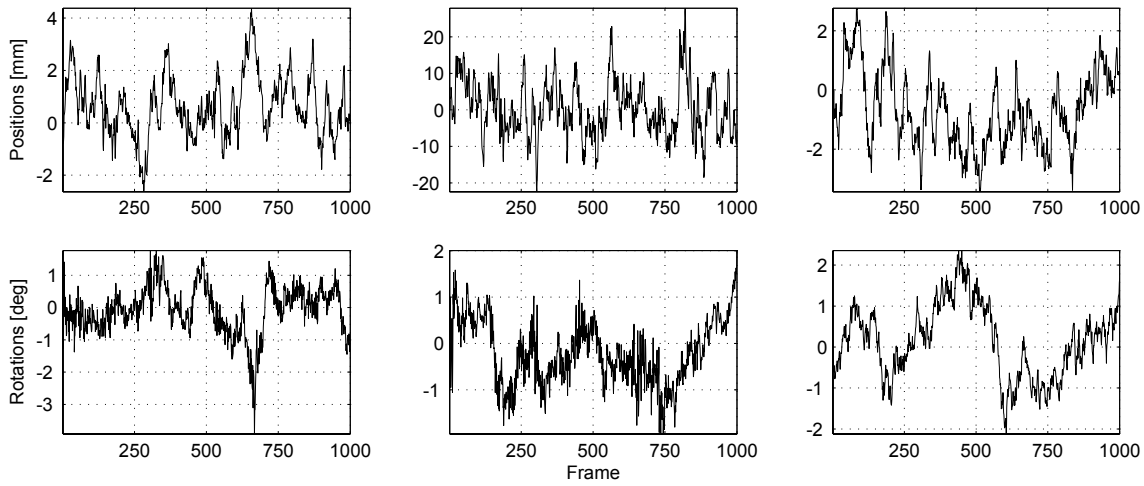

Figure 3. Position and orientation errors

4 Initialization and Motion Prediction

4.1 Global pose initialization

For the initialization phase a feature matching approach is used to estimate the initial position of the object. In order to detect the object in the current image a reference image is required. Using the Harris corner detector[5] feature points are extracted from the reference image and subsequently matched to features found in the current image. This is accomplished by calculating the normalized cross-correlation between the neighborhoods of all detected features. However since normalized cross-correlation is not rotation-invariant, multiple reference images are created from the image provided by the user by incrementally rotating it by 40 degrees. During the initialization phase the current image is matched against all reference images using the following matching strategy:

For a given point p in the reference image the point q in the current image with the highest cross-correlation is selected. If, conversely, the point which has the highest correlation with q is p, the point pair (p, q) is considered a potential match. Afterwards, in order to robustly remove wrong matches, the RANSAC [2] algorithm is used, together with a 2D-homography model.

Once the RANSAC algorithm terminates, all the remaining inlier features are used to estimate the best homography using a least squares approximation. The transformation obtained from the reference image with the highest number of inliers is finally applied to the control points of the contour spline model, in order to obtain an initial estimate of the contour position.

4.2 Trajectory prediction and check

In order to detect a tracking failure without the need for absolute pose information at every timestep, we employ a standard steady-state Kalman Filter that incorporates a "reasonable" noisy kinematic model of the trajectory, known as Discrete White Noise Acceleration Model [7]. For this system, the steady-state estimation filter (or *alpha-beta* filter) is employed, with suitable covariance parameters for each state variable. In order to check the trajectory, an upper threshold on the measurement residuals between the predicted variables and the estimated values (according to the CCD algorithm) is set, and if the error exceeds the threshold, the measured trajectory is considered as unreliable and reinitialized. The reader is referred to [7] for more details on the topic.

5 Experimental Results and Conclusion

In order to evaluate the performance of the tracking system, a simulated sequence was generated by superimposing a fictitious planar object onto a moving background sequence taken from an office scenario. The object moves w.r.t. the camera by following a second-order random walk motion model, obtained by adding Gaussian acceleration noise to a linear AR state system, with independent dynamics for each dof. In Fig. 2 some frames taken from the simulation are shown, and both the estimated contour and 3D object frame positions are displayed.

The results of the tracking are shown in Fig. 3, where the 6 trajectory estimation errors w.r.t. the real values are shown; the translation errors are measured in [*mm*], and the orientation errors are expressed by using the equivalent

Figure 4. Results of two real-time tracking experiments

axis-angle representation, where the three components of the rotation error vector are given in [*deg*]. The *rms* estimation error over the whole sequence is lower than 2*mm* for the 2 planar translations, 7*mm* for the depth component, and 1*deg* for the rotations.

Subsequently, we tested the tracking system on real-life scenarios, and in Fig. 4 a few pictures from two real-time experiments are shown. The frames show the online tracking performance after the initialization under different conditions, such as partial occlusion, cluttered background, changing lighting, and widely different object positions; almost no tracking loss occurred during the sequence. For all the experiments, a standard webcam with a resolution of 640x480 has been employed, without any preprocessing of the input images. The tracking system runs in real-time on a 3 GHz PC, and the processing speed of the system alone permits a frame rate of more than 60 fps.

A new version of the system is currently under development, which includes multiple (3D) contour models, high dimensional shape spaces, and more general global search techniques for the initialization module.

References

[1] A. Blake and M. Isard. *Active Contours: The Application of Techniques from Graphics,Vision,Control Theory and Statistics to Visual Tracking of Shapes in Motion*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1998.

[2] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, 1981.

[3] R. Hanek and M. Beetz. The contracting curve density algorithm: Fitting parametric curve models to images using local self-adapting separation criteria. *International Journal of Computer Vision (IJCV)*, 59(3):233–258, 2004.

[4] R. Hanek, T. Schmitt, S. Buck, and M. Beetz. Fast Image-based Object Localization in Natural Scenes. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2002*, Lausanne, pages 116–122, 2002.

[5] C. J. Harris and M. Stephens. A combined corner and edge detector. In *Proc. 4th Alvey Vision Conference, Manchester*, pages 147–151, 1988.

[6] M. Isard and A. Blake. Condensation – conditional density propagation for visual tracking. *International Journal of Computer Vision (IJCV)*, 29(1):5–28, 1998.

[7] T. K. Y. Bar-Shalom, X. Rong Li. *Estimation with applications to Tracking and Navigation*. John Wiley and Sons, Inc., 2001.