IEEE Int. Conf. on Robotics and Automation, Barcelona, Spain, April 2005

Robust Real-Time Visual Tracking:
Comparison, Theoretical Analysis and Performance Evaluation

Andrew I. Comport*, Danica Kragic§, Éric Marchand*, François Chaumette*

*IRISA - INRIA
Campus de Beaulieu, Rennes, France

§NADA - CAS/CVAP
Royal Institute of Technology, Stockholm, Sweden

***Abstract*— In this paper, two real-time pose tracking algorithms for rigid objects are compared. Both methods are 3D-model based and are capable of calculating the pose between the camera and an object with a monocular vision system. Here, special consideration has been put into defining and evaluating different performance criteria such as computational efficiency, accuracy and robustness. Both methods are described and a unifying framework is derived. The main advantage of both algorithms lie in their real-time capabilities (on standard hardware) whilst being robust to miss-tracking, occlusion and changes in illumination.**

***Index Terms*— Model-based tracking, real-time, robustness.**

I. Introduction

Methods for real-time 3D tracking of objects using vision sensors have recently matured to the point where they are rapidly becoming good enough to be used in real-world applications ranging from robotics to augmented reality and medical applications [12], [5], [10], [13], [3]. In this paper, the problem will be restricted to model-based 3D tracking algorithms where the tracking is considered as a 3D localization issue, handled using *Full-scale non-linear optimization techniques* as proposed more than 10 years ago by David Lowe [12]. Such approaches have proved to be very efficient and can now operate in real-time. Although a number of different techniques exist, most of the reported results are similar.

This paper addresses two recent formulations of such algorithms. The first method considered has been proposed by Drummond and Cipolla [5] and is based on Lie Algebra. The second method has been proposed in [3] and is based on the virtual visual servoing technique (VVS). The two methods are based on optimization and have been demonstrated as robust in real-time settings. It should be noted that even though the formulations are very different, they remain full scale non-linear pose computation algorithms with a 1D search along the projected edge normal in subsequent frames, as well as robust M-estimation.

Since an image stream contains a practically infinite amount of information it is necessary to further sample the image measures so as to obtain pertinent information about the position and orientation of a target object. In computer vision literature geometric primitives considered for the estimation are often points [7], [4], contours or points on the contours [12], [3], [5], segments, straight lines, conics, cylindrical objects, or a combination of these. The methods compared in this paper consider a redundant set of distance measures resulting from an efficient 1D search path. Non-linear minimization is then performed over a set of redundant distance measures. The methods are accurate since distances are considered in a many-to-one mapping between the distances and the object's pose thus allowing averaging of inherent noise across the object evenly.

Real world tracking systems need to be robust to occlusion, miss-tracking and image noise. The main advantage of the model-based methods is that the knowledge about the scene (the implicit 3D information) allows improvement of robustness and performance by estimating the 3D movement of the object and acts to reduce the effects of outlier data introduced in the tracking process. Both methods implement outlier rejection using M-estimators [8] by considering iteratively re-weighted least square(IRLS) minimization techniques.

Let it be noted that the Jacobians or interaction matrices that link the estimated camera velocity to the variation of a distance in the image are derived using different methods and a set of unifying equations are presented in this paper. From now on, Drummond and Cipola's method [5] will be referred to as Method 1 and the VVS method will be referred to as Method 2. In the remainder of this paper, Section II presents an overview of both approaches. Section III presents the Lie Algebra-based tracking approach while Section IV presents Virtual visual servoing one. In Section V, a unification of both approaches as well as the main differences are presented. Finally, in Section VI, experimental results are demonstrated.

II. Robust tracking based on non-linear optimization techniques

Full-scale non-linear optimization techniques consist of minimizing the error between the observation and the forward-projection of the model. In this case, minimization is handled using numerical iterative algorithms such as Newton-Raphson or Levenberg-Marquardt. The main advantage of these approaches is their accuracy. The main drawback is that they may be subject to local minima and, worse, divergence. Both approaches considered in this paper rely on a non-linear optimization approach.

To illustrate the principle, consider the case of an object with various 3D features $\mathbf{S}$ ($^o\mathbf{S}$ represents the 3D parameters of these features in the object frame). A camera reference frame is defined with its pose relative to the object frame defined by $\mathbf{r}$ representing the position and orientation of the camera relative to the object. The homogeneous transformation matrix representing

this pose is denoted:

$$^{c}\mathbf{M_o} = \begin{bmatrix} ^{c}\mathbf{R_o} & ^{c}\mathbf{t_o} \\ \mathbf{0_3} & 1 \end{bmatrix}, \quad (1)$$

where $^{c}\mathbf{R_o}$ and $^{c}\mathbf{t_o}$ are the rotation matrix and translation vector between the camera frame c and the object frame o. These homogeneous transformations belong to the 6-dimensional Lie Group of rigid body motions in $SE(3)$.

The goal of the pose computation problem is to estimate the extrinsic parameters $\mathbf{r}$ by minimizing the error Δ between the observed data $\mathbf{s_d}$ (usually the position of a set of features in the image) and the position $\mathbf{s}$ of the same features computed by forward-projection according to the current extrinsic and intrinsic parameters:

$$\Delta = \mathbf{s}(\mathbf{r}) - \mathbf{s_d} = \mathbf{pr}_{\xi}(\mathbf{r},^{\mathbf{o}}\mathbf{S}) - \mathbf{s_d}, \quad (2)$$

where $pr_{\xi}(\mathbf{r},^{\mathbf{o}}\mathbf{S})$ is the projection model according to the camera pose $\mathbf{r}$ and the intrinsic parameters ξ. It is supposed here that intrinsic parameters ξ are available but it is also possible, using the same approach, to estimate these parameters.

Considering that $\mathbf{s_d}$ is computed (from the image) with sufficient precision is an important assumption. Robust M-estimation is used in both methods to reduce the effect of miss-tracking and occlusion in the image. M-estimators can be considered as a more general form of maximum likelihood estimators [8]. They are more general because they permit the use of different minimization functions not necessarily corresponding to normally distributed data. Many functions have been proposed in the literature which allow uncertain measures to be less likely considered and in some cases completely rejected. The objective function is therefore modified to reduce the sensitivity to outliers. The robust optimization problem is then given by:

$$\Delta_{\mathcal{R}} = \rho\Big(\mathbf{s}(\mathbf{r}) - \mathbf{s_d}\Big), \quad (3)$$

where $\rho(u)$ is a robust function [8] that grows subquadratically and is monotonically nondecreasing with increasing $|u|$. Both approaches consider a robust minimization.

III. Lie group formulation of the tracking

The derivation of the image Jacobian for Method 1 is based on simple point features in 3D and distances to the projection of these points in 2D. The velocity field is defined from Lie Group generators which are projected in the direction of the contour normal so as to represent different rigid motion parameters as components of a distance (see Figure 1).

A. Interaction Matrix - Distance to Point

For the case of a point in 3D, a projection matrix is defined as $\mathbf{P} = \mathbf{KM}$ in the coordinate system of the structure, where $\mathbf{K}$ is composed of the intrinsic camera parameters and $\mathbf{M}$ is composed of the extrinsic pose parameters. The projective coordinates of a point are then given by:

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \mathbf{P} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \quad (4)$$

where the 2D point $\mathbf{p} = (\mathbf{x}, \mathbf{y})$, corresponding to a 3D point, in image pixel coordinates coordinates is given by $x = (u/w)$ and $y = (v/w)$

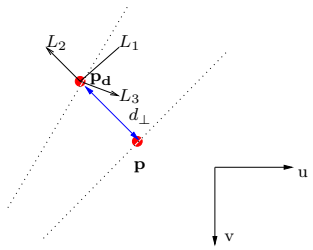


Fig. 1. Distance of a point to a point

In order to perform pose estimation, the derivative of $SE(3)$ corresponding to the tangent velocity vector space or Lie Algebra is considered. In this first method, the 4×4 basis matrices of the Lie Algebra are used explicitly. These basis matrices are called generators and are chosen in a standard way to represent translations in the x, y and z directions along with rotations around the x, y and z axes. These six generators are given in [5].

The Lie Group of displacements is related to the Lie Algebra of velocities via the exponential map, so that:

$$\mathbf{M} = \mathbf{exp}(\alpha_{\mathbf{i}}\mathbf{G_i}), \quad (5)$$

where each α_i corresponds to an element of the kinematic screw or twist representing the inter-frame transformations. The motion in the image is related to the twist in 3D by taking the partial derivative of projective image coordinates with respect to the i^{th} generating motion:

$$\begin{bmatrix} u_i^{'} \\ v_i^{'} \\ w_i^{'} \end{bmatrix} = \mathbf{PG_i} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \quad (6)$$

with the motion in pixels being equivalent to the well known optical flow equation as:

$$\mathbf{l_i} = \begin{bmatrix} \dot{x}_i \\ \dot{y}_i \end{bmatrix} = \begin{bmatrix} \frac{u_i^{'}}{w_i} - \frac{u_i w_i^{'}}{w_i^2} \\ \frac{v_i^{'}}{w_i} - \frac{v_i w_i^{'}}{w_i^2} \end{bmatrix}, \quad (7)$$

giving the motion in image coordinates.

Continuing with the determination of the inter-frame movement, the different generating motions are projected in the direction of the normal of the contour as:

$$f_i = \mathbf{l_i}.\widehat{\mathbf{n}}, \quad (8)$$

where $\widehat{\mathbf{n}}$ is the normal direction.

B. Robust Minimization

Computing the motion is performed by solving a weighted least-squares algorithm as follows:

$$v_i = \sum_{\xi} s(d^{\xi}) d^{\xi} f_i^{\xi}, \quad (9)$$

$$C_{ij} = \sum_{\xi} s(d^{\xi}) f_i^{\xi} f_j^{\xi}, \quad (10)$$

$$\alpha_i = C_{ij}^{-1} v_j, \quad (11)$$

where d^{ξ} is a distance measured normal to a contour in the image and α_i is an estimated velocity corresponding

to one of the six basis generators $\mathbf{G_i}$. Here, the f_i represent elements of the interaction matrix for a distance to a point and $s(d^{\xi})$ is a robust weighting function [8][5].

Rodrigues' formula is then used to map, exponentially, the estimated velocities α_i to the corresponding instantaneous displacement allowing the pose to be updated. To apply the update to the displacement between the object and camera, the exponential map is applied using pose matrices resulting in:

$$^{c}\mathbf{M_{o,t+1}} = ^{\mathbf{c}}\mathbf{M_{o,t}}\mathbf{exp}\left(\sum_{\mathbf{i}}\alpha_{\mathbf{i}}\mathbf{G_i}\right) \tag{12}$$

IV. Virtual visual servoing formulation

The derivation of the interaction matrix for Method 2 is based on the distance between projected 3D features such as lines, circles, cylinders, etc..and 2D points. The velocity of a virtual camera is then related to the velocity of these distances in the image.

A. Interaction Matrix - Distance to Line

In [6] a general technique is given for determining the interaction matrix of any geometric feature, including the projection of the feature's contour in the image along with its differential relation to the kinematic screw of the camera. Following this general methodology a simple line feature has been considered.

The derivation of the interaction matrix for the velocity parameters linking the variation of the distance between a fixed point and a moving line to the virtual camera motion is modeled by first considering the distance in 2D between a point and a line. In Figure 2, $\mathbf{p}$ is the tracked point feature position and $\mathbf{l}(\mathbf{r})$ is the current line feature position.

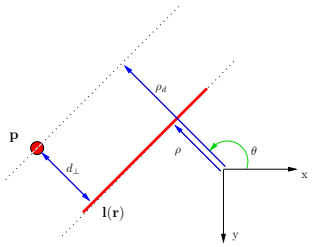


Fig. 2. Distance of a point to a line

The distance feature from a line is given by:

$$d_l = d_\perp(\mathbf{p}, \mathbf{l}(\mathbf{r})) = \rho(\mathbf{l}(\mathbf{r})) - \rho_{\mathbf{d}}, \tag{13}$$

where

$$\rho_d = x_d\cos\theta + y_d sin\theta, \tag{14}$$

with x_d and y_d being the coordinates of the tracked point. Thus,

$$\dot{d}_l = \dot{\rho} - \dot{\rho_d} = \dot{\rho} + \alpha\dot{\theta}, \tag{15}$$

where $\alpha = x_d\sin\theta - y_d\cos\theta$. Introducing the respective interaction vectors for a line into (15) gives $\mathbf{L}_{d_l} = \mathbf{L}_\rho + \alpha\mathbf{L}_\theta$. The interaction vector related to d_l can be thus derived from the interaction matrix related to a straight line given by (see [6] for its complete derivation):

$$\begin{aligned}\mathbf{L}_\theta &= \begin{bmatrix}\lambda_\theta\cos\theta & \lambda_\theta\sin\theta & -\lambda_\theta\rho & \rho\cos\theta & -\rho\sin\theta & -1\end{bmatrix}\\ \mathbf{L}_\rho &= \begin{bmatrix}\lambda_\rho\cos\theta & \lambda_\rho\sin\theta & -\lambda_\rho\rho & (1+\rho^2)\sin\theta & -(1+\rho^2)\cos\theta & 0\end{bmatrix}\end{aligned} \tag{16}$$

where $\lambda_\theta = (A_2\sin\theta - B_2\cos\theta)/D_2$, $\lambda_\rho = (A_2\rho\cos\theta + B_2\rho\sin\theta + C_2)/D_2$, and $A_2X + B_2Y + C_2Z + D_2 = 0$ is the equation of a 3D plane which the line belongs to.

From (15) and (16) the following is obtained:

$$\mathbf{L}_{d_l} = \begin{bmatrix}\lambda_{d_l}\cos\theta \\ \lambda_{d_l}\sin\theta \\ -\lambda_{d_l}\rho \\ (1+\rho^2)\sin\theta - \alpha\rho\cos\theta \\ -(1+\rho^2)\cos\theta - \alpha\rho\sin\theta \\ -\alpha\end{bmatrix}^T, \tag{17}$$

where $\lambda_{d_l} = \lambda_\rho + \alpha\lambda_\theta$.

Let it be noted that the case of a distance between a point and the projection of an ellipse has been considered in [3] and the case of a distance to a cylinder is very similar to the case of a line.

B. Robust minimization

In Method 2, minimization of the distances is considered as a dual problem of 2D visual servoing [6], [9]. In visual servoing, the goal is to move a camera in order to observe an object at a given position in the image. This is achieved by minimizing the error between a desired state of the image features $\mathbf{s_d}$ and the current state $\mathbf{s}$. If the vector of visual features is well chosen, there is only one final position of the camera that allows this minimization to be achieved. Alternatively for pose computation, a virtual camera is moved (initially at $\mathbf{r_i}$) using a visual servoing control law in order to minimize this error Δ. This virtual camera finally reaches the position $\mathbf{r_d}$ which minimizes this error ($\mathbf{r_d}$ will be the real camera pose).

The objective of the control scheme is to minimize the objective function given in (3). This new objective is incorporated into the control law in the form of a weight which is given to specify a confidence in each feature location. Thus, the error to be regulated to 0 is defined as:

$$\mathbf{e} = \mathbf{D}(\mathbf{s}(\mathbf{r}) - \mathbf{s_d}), \tag{18}$$

where $\mathbf{D}$ is a diagonal weighting matrix given by $\mathbf{D} = \mathbf{diag}(\mathbf{w_1}, \ldots, \mathbf{w_k})$.

A simple control law can be designed to try to ensure an exponential decoupled decrease of $\mathbf{e}$ around the desired position $\mathbf{s}^*$ (see [3] for more details). The control law is given by:

$$\mathbf{v} = -\lambda(\widehat{\mathbf{D}\mathbf{L}_\mathbf{s}})^{+}\mathbf{D}\big(\mathbf{s}(\mathbf{r}) - \mathbf{s}^*\big), \tag{19}$$

where $\mathbf{v}$ is the virtual camera velocity.

A classical approach in visual servoing considers $\widehat{\mathbf{D}\mathbf{L}_\mathbf{s}}$ to be constant and it is calculated from the desired depth $\mathbf{Z}^*$ and the desired value of the features $\mathbf{s}^*$. In the VVS case, the desired depth is unknown but the initial value of $\mathbf{Z_i}$ is available, thus $(\widehat{\mathbf{D}\mathbf{L}_\mathbf{s}})^{+}$ can be defined as: $(\widehat{\mathbf{D}\mathbf{L}_\mathbf{s}})^{+} = \mathbf{L}^{+}(\mathbf{s_i}, \mathbf{Z_i})$.

As is done in Method 1, Rodrigues' formula is then used to map the velocity vector $\mathbf{v}$ to its corresponding instantaneous displacement allowing the pose to be updated. To apply the update to the displacement between the object and camera, the exponential map is applied using homogeneous matrices resulting in:

$$^{c}\mathbf{M}_o^{t+1} = ^{c}\mathbf{M}_o^{t} exp(\mathbf{v}) \tag{20}$$

V. Algorithms Unification and Comparison

In this section it will be demonstrated that although the formulation of each method is very different, both algorithms can be unified in a common mathematical framework. This common framework, in turn, provides a basis from which various differences may be explained. Following the unification, these differences are described.

A. Notations

First of all, the equivalence between the notation of Method 1 and the matrix notation of Method 2 is established. The velocities α_i to be estimated in the first method are elements of the kinematic screw and can be written as $\mathbf{v} = (\alpha_\mathbf{1} \dots \alpha_\mathbf{6})$. The index i is chosen to index a feature in place of ξ in the first method. The distances are defined similarly giving $d_l = d_\xi$ with error vector defined as $\mathbf{s} = (\mathbf{d_1}, \dots, \mathbf{d_n})$, where n is the number of features. The weights $s(d^\xi)$ are considered equivalent to the weights w_i. As will be demonstrated in Section V-B, the f_i from (8) are equivalent to elements of the interaction matrix given in (17).

B. Interaction Matrices Derivation

In this section it is shown that the principle difference between the first approach and the second is in the method for deriving the interaction matrix or image Jacobian. Since the second method is centered around the modeling of the interaction matrix it is easily adapted to be used as a unifying framework.

In Method 1 components of the interaction matrix were determined individually by the use of Lie Group generators and homogeneous point projection matrices. In this case points were sampled along the contours in 3D. Alternatively, in Method 2 the interaction matrix was calculated via a direct projection of higher level 3D features onto the image. In this case the sampling of the contours occurs along the 2D projection of the contour in the image. Therefore, if the first method is considered in this common framework, the 3D model can be considered as a collection of 3D points.

The derivation of the interaction matrix of a point in the first method can be written in its matrix form by considering each $\mathbf{l_i}$ from (7) to correspond to a column of the interaction matrix for a point so that the interaction matrix for a point is equivalent to $\mathbf{L_p} = [\mathbf{l_1}, \dots, \mathbf{l_6}]$, equivalent to the well known interaction matrix for a point given as:

$$\mathbf{L_p} = \begin{bmatrix} -Z^{-1} & 0 & xZ^{-1} & xy & -(1+x^2) & y \\ 0 & -Z^{-1} & yZ^{-1} & 1+y^2 & -xy & -x \end{bmatrix}$$

The dot product operator that appears in (9) is equivalent to projecting each point coordinate's interaction vector in the direction of the normal as:

$$\mathbf{L}_{dp} = \cos\theta \mathbf{L}_x + \sin\theta \mathbf{L}_y \tag{21}$$

giving the interaction matrix of a distance to a projected point as:

$$\mathbf{L_{d_p}} = \begin{bmatrix} -Z^{-1}\cos\theta \\ -Z^{-1}\sin\theta \\ Z^{-1}x\cos\theta + Z^{-1}y\sin\theta \\ xy\cos\theta + (1+y^2)\sin\theta \\ -(1+x^2)\cos\theta - xy\sin\theta \\ y\cos\theta - x\sin\theta \end{bmatrix}^\mathbf{T}, \tag{22}$$

where the subscript d_p refers to a distance to a point as opposed to a distance to a line in equation (17).

This can be proven as being mathematically equivalent to the distance to line interaction matrix (17) by rewriting equation (15) to eliminate ρ by using the equation of a line (14).

$$d_l = (x - x^*)\cos\theta + (y - y^*)\sin\theta. \tag{23}$$

Deriving (23) with respect to time gives the same as (21):

$$\dot{d}_l = \dot{x}\cos\theta + \dot{y}\sin\theta, \tag{24}$$

Therefore both methods can be considered as equivalent mathematically yet slightly different in terms of modeling the interaction matrix. Furthermore, it has been shown that the two methods can be considered within a unifying framework.

The notable difference in the two derivations, although, lies in the computational complexity of the algorithm. This is due to the fact that for Method 1 sampling of the contour occurs in 3D whereas in Method 2 this is done in 2D. Furthermore, each method does a conversion from meters to pixels in a different way. In Method 1 the 3D points are sampled, projected and then converted to pixels, whereas, in Method 2 the projected contours are both converted to pixels and sampled in 2D. It can be shown that the computational complexity of Method 1 for the projection for all sampled points onto the image is order $O(n)$ and for the intrinsic parameter conversion from meters to pixels is also $O(n)$ where n is the number of tracked points along the contour. In Method 2 both extrinsic projection and intrinsic conversion are performed in order $O(m)$, where m is the number of higher level features and where $m << n$.

C. Robust Minimization

Finally, the linearized least squares minimization in (9) of Method 1 can be rewritten in the common form:

$$\mathbf{v} = (\mathbf{DL})^+ \mathbf{D}(\mathbf{s}(\mathbf{r}) - \mathbf{s_d}), \tag{25}$$

with the pseudo-inverse $(\mathbf{DL})^+$ being equivalent to the computation of the individual terms C_{ij} in equation (9). It can be noted in this case that the gain $\lambda = 1$.

It can be seen that the second method includes an iterative calculation, in equation (19), with a time constant determined by the gain λ, whereas in Method 1 only a single iteration is performed for each frame of the video. It can be noted that the iterative process is slightly more time consuming, however, it allows greater convergence upon the global minimum which ensures greater precision.

For the implementation of a robust M-estimator, an iterative re-weighted least squares (IRLS) approach is

used in both approaches to handle outliers, however, at this point the effect of the differences is more significant. In Method 1, since a single iteration is performed for each frame, IRLS is performed over sequential frames. Since the weights for each distance are calculated iteratively it is necessary, in this case, to keep a one-to-one correspondence between sampled points across frames so as to be always estimating the same weight for the same point. This implies that edge re-sampling is not done at each new frame and subsequently that the pose update using the exponential map is only done after convergence across frames. In the interpretation of the algorithm in this paper, tracking was performed at each step, however, in this case it is easy to show that the M-estimator does not work without convergence. These complications could explain why Method 1 considered other rejection rules to deal with particular problems as described in [5].

In Method 2, since minimization procedes iteratively for each frame, weights and scale (standard deviation of the inliers data) are recomputed iteratively (see [3] for details). This allows adaptive and precise rejection of outliers in the data. Furthermore, the edge sampling across frames can be modified online, due to self-occlusion for example, without any problem. Dealing with the implementation of the M-estimator, the two methods also feature minor differences with the choice of influence function (McLure in Method 1 and Tukey in Method 2).

Once again if the complexity is re-considered by using an iterative approach for both methods (which improves Method 1's behavior) the order of complexity becomes $O(2ne^{-\lambda t})$ and $O(2me^{-\lambda t})$ for Method 1 and 2 respectively.

D. Low level issue

When dealing with image processing, the normal displacements were evaluated along the projection of the object model contours. Both approaches are similar and can be implemented with convolution efficiency leading to real-time computation. The process consists in searching for the corresponding p^{t+1} in image I^{t+1} of each point p^t. A 1D search interval $\{Q_j, j \in [-J, J]\}$ is determined in the direction δ of the normal to the contour. For each point p^t and for each entire position Q_j lying in the direction δ a criterion ζ_j is computed.

In Method 1 Q_j^* corresponds to the nearest intensity discontinuity along the edge normal. In this case $\zeta_j = \mid I^{t+1}_{\nu(Q_j)} * M \mid$ where M is an edge detection mask. In this case the search starts at a point p^t and tests all the points Q_j along the normal until $\zeta_j > \xi$ (where ξ is a given threshold). In other words, the closest point above a certain threshold is selected.

In Method 2, the Moving Edge algorithm [1] is considered. The new position p^{t+1} is given by the maximum likelihood of a contour in the search path:

$$Q^{j^*} = \arg \max_{j \in [-J,J]} \zeta_j \text{ with } \zeta_j = \mid I^t_{\nu(p^t)} * M_\delta + I^{t+1}_{\nu(Q_j)} * M_\delta \mid$$

where $\nu(.)$ is the neighborhood of the considered pixel and M_δ are pre-computed mask function of the orientation of the contour. Therefore this method chooses the maximum likelihood of a contour existing along a fixed distance to the normal. Method 2 only finds edges with the same orientation and with similar contrast which improves accuracy and subsequent efficiency of the tracking as opposed to choosing any edge in the path with any orientation.

VI. Results

In the following section several experiments on each method are presented. The implementation of the code was made exactly the same for each method except for essential code in the time-critical inner loop. In the experiments presented, images are acquired using a commercial digital camera. The object used in the experiments was an 'Uncle Ben's' rice packet. The reason for choosing this object is that its textural properties give rise to a number of outliers. Several different movements were performed including large rotations and translations. In the experiments, tracking is always performed at below frame rate.

The SIFT point matching method [11] was used along with reference images of the object and a linear pose computation method to perform an automatic initialization of the pose in the first image. An example of the initialization step can be seen in Figure 3.

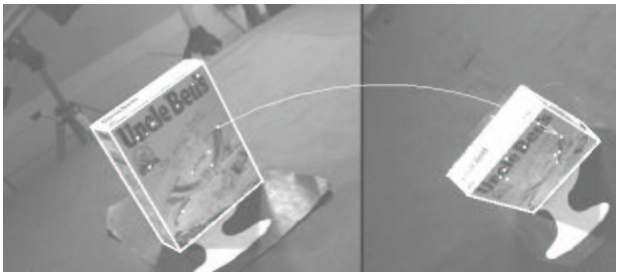


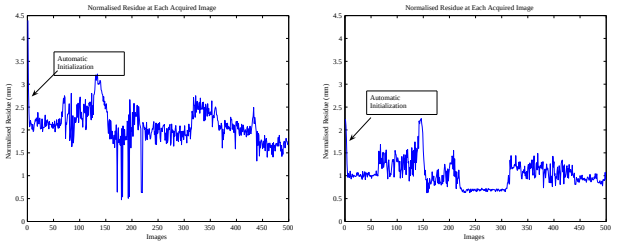
Fig. 3. Automatic initialization of the pose in the first image for two different starting positions. Reference image on the left with matched point s indicated on each image.

A. Iterative vs non-iterative minimization

The goal of the first experiment was to analyze the advantages of using a non-iterative (Method 1) versus an iterative minimization scheme (Method 2). The accuracy without iterating gives a higher normalized averaged error across the sequence than when iterations are performed (see Figure 4). Obviously iteratively minimizing for each frame obtains a more precise minimum of the objective function. Furthermore the larger deviation of error seen in the plot for Method 1 can also be explained by the failure of the M-estimator which does not obtain an accurate scale estimate (standard deviation of the inliers data) without iterating. This could be chosen manually for each sequence, however, automatic determination of scale is preferred.

B. Low level

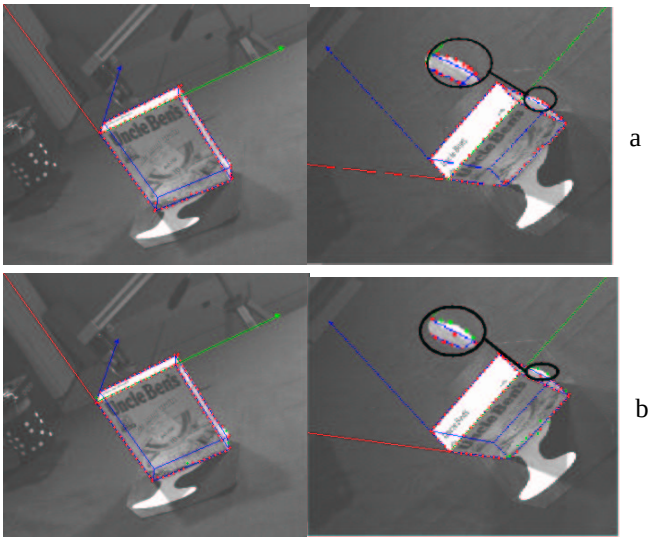
In the second experiment, we have compared different low level feature detection algorithms. Taking the maximum likelihood value in the direction of the normal (Method 2) worked better than taking the closest likelihood over a certain threshold (Method 1) due to the fact that Method 1 is very sensible to the chosen detection threshold. Furthermore, propagating the weights (as



(a) (b)

Fig. 4. Comparison of error norm over a 500 image sequence. a) Method 1 without iterating for each image and b) Method 2 using an iterative scheme with a gain $\lambda = 0.7$. The iterative method provides better minimization along with a better implementation of the IRLS algorithm for outliers rejection.

considered in [2]) was needed for the Uncle-Ben object because the inner-contours of the object have a highly different contrast to the exterior edges meaning that a fixed threshold would need to be tuned to obtain the optimal balance between inner and outer contours. In Figure 5, a failure case is shown for the use of Method 1 compared to the use of Method 2.



a

b

Fig. 5. The same images of a sequence using each method. Projected model is drawn in blue and the object axes are drawn in red, green and blue. (a) Method 1 with tracked points drawn in red, miss-tracking can be observed at the bottom right hand corner of the box as the M-estimator is not effective without iterating. (b) Method 2 with the M-estimator rejecting outliers and rejected points in green.

C. Computational Efficiency

The computational efficiency was measured by timing the critical inner loop for execution time. Since the two methods are so similar, a direct comparison of the computational time is not possible and without the author's implementation of Method 1's code this seems difficult to compare. It is more interesting to present the computational times for these sorts of algorithms in order to show where additional computation improvement can be made. The pie chart in Figure 6 is the result of an analysis of the computation time occupied by the different functions. The current graph is for the implementation of Method 2, however, the graph for Method 1 is practically the same except for some extra computation time spent on the computation of non-optimized code for the multiplication of projection matrices and generator matrices. It can clearly be seen that the major factor for computational efficiency is the convolution for determining edge locations. Coming in second is the time required to compute the pseudo-inverse and multiplication of the interaction matrix in the control law. Thus it is preferable to use the interaction matrix in its fully derived form and not calculate the intermediary steps of projection and meters-to-pixel for each generator at each iteration.

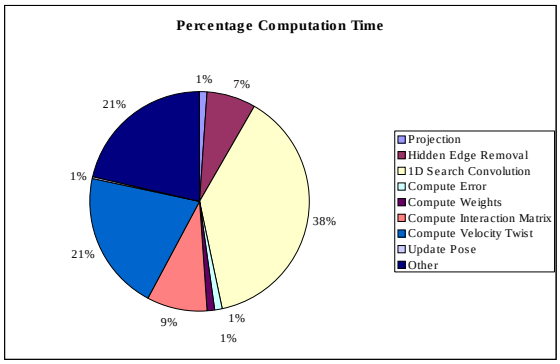


Fig. 6. Percentage computation time of different parts of the algorithm

VII. Conclusion

This paper has presented and compared two real-time tracking algorithms. A unifying mathematical framework has been described showing clearly the principle differences of the algorithms. In particular, a clear formulation using an interaction matrix representation has been given to unify the derivation of any distance to contour type image feature. The criteria of accuracy, efficiency and robustness have been addressed in theory and confirmed in practice. It has been shown that it is necessary to iterate for both improved precision and accurate outlier rejection. Furthermore, tuning parameters such as a contour detection threshold and a robust scale parameters are eliminated in this approach.

References

[1] P. Bouthemy. A maximum likelihood framework for determining moving edges. *IEEE Trans. on PAMI*, 11(5):499–511, May 1989.

[2] A.I. Comport, E. Marchand, and F. Chaumette. A real-time tracker for markerless augmented reality. In *ACM/IEEE Int. Symp. on Mixed and Augmented Reality*, pp. 36–45, Tokyo, Oct. 2003.

[3] A.I. Comport, E. Marchand, and F. Chaumette. Robust model-based tracking for robot vision. In *IEEE/RSJ Int. Conf. on IROS'04*, vol. 1, pp. 692–697, Sendai, Sept. 2004.

[4] D. Dementhon and L. Davis. Model-based object pose in 25 lines of codes. *Int. J. of Computer Vision*, 15:123–141, 1995.

[5] T. Drummond and R. Cipolla. Real-time visual tracking of complex structures. *IEEE T.. on PAMI*, 27(7):932–946, July 2002.

[6] B. Espiau, F. Chaumette, and P. Rives. A new approach to visual servoing in robotics. *IEEE Trans. on Robotics and Automation*, 8(3):313–326, June 1992.

[7] R. Haralick, H. Joo, C. Lee, X. Zhuang, V Vaidya, and M. Kim. Pose estimation from corresponding point data. *IEEE Trans on Systems, Man and Cybernetics*, 19(6):1426–1445, Nov. 1989.

[8] P.-J. Huber. *Robust Statistics*. Wiler, New York, 1981.

[9] S. Hutchinson, G. Hager, and P. Corke. A tutorial on visual servo control. *IEEE Trans. on Robotics and Automation*, 12(5):651–670, Oct. 1996.

[10] D. Kragic and H.I. Christensen. Confluence of parameters in model based tracking. In *IEEE ICRA'03*, vol. 4, pp. 3485–3490, Taipe, Sept. 2003.

[11] D. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. of Computer Vision*, 60(2):91–110, 2004.

[12] D.G. Lowe. Fitting parameterized three-dimensional models to images. *IEEE Trans. on PAMI*, 13(5):441–450, May 1991.

[13] L. Vacchetti, V. Lepetit, and P. Fua. Stable 3–d tracking in real-time using integrated context information. In *CVPR'03*, vol. 2, pp. 241–248, Madison, WI, June 2003.