Accurate 3D Tracking of Rigid Objects with Occlusion Using Active Appearance Models

Pradit Mittrapiyanuruk[1], Guilherme N.DeSouza[2], Avinash C. Kak[1]

[1] *Robot vision Laboratory, School of Electrical and Computer Engineering, Purdue Univerisy, USA*
[2] *School of Electrical, Electronic & Computer Engineering, The University of Western Australia*
mitrapiy@purdue.edu, gdesouza@ee.uwa.edu.au, kak@ecn.purdue.edu

Abstract

In this paper we present a new method for tracking rigid objects using a modified version of the Active Appearance Model. Unlike most of the other appearance-based methods in the literature, such as [3,5,6,9,11], our method allows for both partial and self occlusion of the objects. We use ground-truth to demonstrate the accuracy of our tracking algorithm. We show that our method can be applied to track moving objects over wide variations in position and orientation of the object – one meter in translation and 140 degrees in rotation – with an accuracy of a few millimeters.

1. Introduction

Tracking an object in 3D space, that is, determining the position and orientation of the object in 6 DOF, plays an important role in many applications. For instance, in many state-of-the-art robotic applications determining the exact pose of a moving object is crucial in order to control the robot end-effector to perform a task on that object.

A classical approach for tracking 3D objects relies on matching geometrical features of the object with the image [1, 2]. However, when scenes are complex, either because the objects are complex or the background is cluttered, extracting and matching features using geometric-based methods can be daunting. As an alternative, several appearance-based methods [3, 5, 9, 11] have been proposed.

One limitation of early implementations of appearance-based methods [5, 11] is in the fact that they could not model the appearance of an object for different viewing angles – which leads to problems of self-occlusion. Since the object appearance in a 2D image depends on the viewing position, tracking a face that moves from left to right profile, for example, could not be done robustly with such methods.

In order to solve this problem, extensions to the appearance based approach have been proposed that use view-based appearance representations [6, 7, 8]. The common idea among these methods is that the appearance of an object is modelled using a set of 2D templates corresponding to the various views of the object. While tracking an object, the algorithm selects a suitable template (or model) to use for the purpose of appearance matching. Since only 2D information about the object is available through the templates, this approach obviously requires model switching. Methods that have been proposed so far for model switching tend to be complicated.

In [6] for example, the original AAM algorithm was modified so that the model parameters could generate many possible views of the object. While this modification was sufficient in the context of the application presented – face recognition – it was constrained by only allowing one degree of freedom in the rotation of the head (pan). This contribution also could not handle partial occlusions.

Another variation of AAM, presented in [4], also constructed a single appearance model obtained from images of the object taken from multiple views. The idea was based on the assumption that there are some common points among these training samples that can be used to align each sample with respect to a common template. However, this assumption is valid only in some special cases such as the case of tracking faces across left-profile to right-profile and it does not apply for irregular and asymmetric objects.

In [3], a different appearance-based method called 3D Morphable Models (3DMMs) was proposed. Since this model carries out a 3D encoding of object shape and texture, problems such as self-occlusion can be properly handled. However, despite the formulation of the method using 3D coordinates, the test and the results presented in [3] apply only to the case of 1DOF in the rotation of the head and, again, there is no provision to deal with partial occlusions.

Finally, in [12], the authors compared 3DMMs to the classical 2D AAMs. They demonstrated how the representational power of 2D AAMs can lead to meaningless instances of the object and how it can be constrained to overcome this problem. But again, they stopped short of demonstrating the usefulness of their method when dealing with radical variations in rotation. Instead, they presented results for only +/- 18 degrees in yaw (pan) and even less than that for pitch and roll.

In this paper, we propose a new method that combines a sparse 3D model with multiple appearance models to accurately estimate the 6 DOF pose of objects. Unlike the range maps in [3], our 3D model is very simple and can be created using a few mouse clicks on a pair of stereo images. However, despite the simplicity of this 3D model, our method provides means to deal with both partial and self occlusion with high accuracy – typically a few millimetres in translation and less than one degree in rotation. Our approach treats self-occlusion and partial occlusion

separately. Issues related to self-occlusion are resolved by using multiple view-specific templates and a simple 3D model, and issues related to partial occlusion resolved by using robust statistics within the appearance based paradigm. This is unlike the 2D work in [16] where both self and partial occlusion are dealt with using robust statistics. Our approach allows for arbitrary rotations of the 3D object.

2. Background

In this section, we will first briefly review the concepts behind Active Appearance Models. Since the method that we propose in section 3 is an extension from our earlier work in [11], we will then provide a quick review of that work in section 2.2.

2.1. Review of Active Appearance Models

In AAM, the appearance model of an object is defined by its shape and texture. The shape model is a set of 2D landmark points represented by a vector: $\mathbf{s} = [x_1 \, y_1 \, ; .. \, x_n \, y_n]^T$, where $(x_i \, y_i)$ are the coordinates of the ith landmark point. A PCA analysis is performed on the training samples to obtain the mean shape $\overline{\mathbf{s}}$ and the shape eigenvector matrix $\mathbf{P_s}$. Hence, an instance of the shape can be expressed in term of the shape parameter vector $\mathbf{b_s}$ as:

$$\mathbf{s} = \overline{\mathbf{s}} + \mathbf{P_s}\mathbf{b_s} \qquad (1)$$

The texture model is defined by the set of intensity values of the pixels lying inside the reference shape $\mathbf{s_{ref}}$. That is, the texture model is the vector $\mathbf{g}=[g_1 \, g_2 \, .. \, g_m]^T$, where g_i is the intensity value of the $\mathbf{x}_i$ pixel inside $\mathbf{s_{ref}}$. Again, a PCA analysis is performed on the training samples and an instance of the texture can be expressed by:

$$\mathbf{g} = \overline{\mathbf{g}} + \mathbf{P_g}\mathbf{b_g} \qquad (2)$$

where $\overline{\mathbf{g}}$ is the mean texture vector, $\mathbf{P_g}$ the texture eigenvector matrix, and $\mathbf{b_g}$ the texture parameter vector.

In order to locate the object in the current image $\mathbf{I}$, the matching algorithm determines the model parameters ($\mathbf{b_s}$, $\mathbf{b_g}$, and $\mathbf{t}_{2D}$) that minimize the difference between the model texture $\mathbf{g}$ and the observed texture $\mathbf{g_o}$. That is

$$(\mathbf{b_s},\mathbf{b_g},\mathbf{t}_{2D}) = \arg\min \sum_i [g(\mathbf{x}_i;\mathbf{b_g}) - g_o(\mathbf{x}_i;\mathbf{b_s},\mathbf{t}_{2D})]^2 \quad (3)$$

where

$$g_o(\mathbf{x}_i) = \mathbf{I}(\mathbf{W}(\mathbf{x}_i;\mathbf{b_s},\mathbf{t}_{2D}))$$

or, for simplicity of notation

$$(\mathbf{b_s},\mathbf{b_g},\mathbf{t}_{2D}) = \arg\min \sum_i [r_i(\mathbf{b_s},\mathbf{b_g},\mathbf{t}_{2D})]^2 \quad (3a)$$

where: r_i is the texture difference, or residue, between the ith pixel of model texture and the corresponding pixel in the current image; $\mathbf{W}(\mathbf{x_i})$ -- by abuse of notation -- is a warping function that relates those two sets of pixels; and $\mathbf{t}_{2D}=[t_x,t_y,\Theta,\lambda]$ are the parameters of a rigid transformation that defines the pose of the object in 2D.

2.2. Tracking 3D rigid objects using AAM -- single view

In [11], we proposed a modification of the AAM approach to track the 6DOF pose of an object – but without occlusions. In that contribution, we represent the shape of an object by the 3D coordinates of its landmark points. That is:

$$\mathbf{S} = \begin{bmatrix} X_1 & ... & X_n \\ Y_1 & ... & Y_n \\ Z_1 & .. & Z_n \end{bmatrix} \qquad (4)$$

As for the texture model, we then extract the intensity values from the pixels lying inside the contour defined by the perspective projection of the landmark points $\mathbf{S}$ in eq.(4). The texture model can then be defined exactly as in eq. (2), and the perspective projections of the 3D landmark points given by:

$$\mathbf{u} = \begin{bmatrix} x_1 & ... & x_n \\ y_1 & ... & y_n \end{bmatrix} = \mathbf{K} * {}^c\mathbf{H_o} * \mathbf{S} \qquad (5)$$

Here, $\mathbf{K}$ and ${}^c\mathbf{H_o}$ represent respectively the intrinsic and the extrinsic parameters of the camera. Where the intrinsic parameters are constant and the extrinsic, ${}^c\mathbf{H_o}$, capture the 6 DOF pose of the object w.r.t the camera, also represented by $\mathbf{t}_{3D}=[t_x, t_y, t_z, r_x, r_y, r_z]$.

In addition to the above, the contribution in [11] also used the following modification of the matching algorithm: it now consisted of determining a set of parameters, $\mathbf{t}_{3D}$ and $\mathbf{b_g}$, that minimize the following equation:

$$\begin{aligned} (\mathbf{b_g},\mathbf{t}_{3D}) &= \arg\min \sum_i [g(\mathbf{x}_i;\mathbf{b_g}) - \mathbf{I}(\mathbf{W}(\mathbf{x}_i;\mathbf{t}_{3D}))]^2 \\ &= \arg\min \sum_i [r_i(\mathbf{b_g},\mathbf{t}_{3D})]^2 \end{aligned} \quad (6)$$

This approach has various advantages over the methods presented in [3], [5], [6], and [12]. For example, vis-à-vis the method presented in [6], we can now accurately estimate the full pose of an object and allow for changes in camera view in all 6DOF (t_x, t_y, t_z, r_x, r_y, and r_z), instead of just in the pan angle. Besides, a Gauss-Newton based minimization algorithm similar to that used in [5] can be applied to minimize (6) as explained in [11].

In terms of 3D approaches, such as those presented in [3] and [12], our method relies on a sparse set of landmark points: roughly 30 landmark points for common real-world objects. This number of landmarks is a few orders of magnitude smaller than the dense set of points used in [3] – more than 70,000 points in that case. Also, this set of 3D landmark points is obtained using a simple tool that was also presented in [11]. This software tool displays two stereo images of the object and guides the human in selecting landmark points in the left image and the corresponding points in the right image. After a few clicks of the mouse, and using simple 3D reconstruction, a 3D model of the object is created. This model is of course greatly more accurate than the one obtained using structure-from-motion as in [12].

A similar method using AAM and a 3D-pose parameterization for $\mathbf{t}_{3D}$ is employed in [9]. However, in their approach the texture parameter $\mathbf{b}g$ is not included in the final parameterization. Instead, they approximate the model

texture using the current observed texture vector, which from our experience produces errors in the AAM matching.

In the next section we present the details of a new AAM-based method for tracking 3D rigid object with occlusions. As we mentioned earlier, this new method extends our previous work in such a way to allow for occlusions: both self and partial.
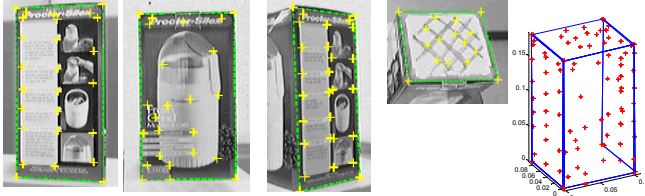

Figure 1. The model of the box: (i) 4 different appearance models with the selected landmark points (yellow crosses); and (ii) the 3D Model from the combined 3D landmark points.

3. Proposed New Method

In order to handle self-occlusion, we employ a multi-view approach similar to the one in [6]. That is, each object is represented by multiple appearance models, each one representing one of the possible views that the object can assume in front of the camera. As for the second type of occlusion, partial occlusion, we employ a robust estimation framework within the matching algorithm.

3.1. Object Model

The complete object model consists of: (i) a set of multiple appearance models; and (ii) a 3D model. In order to construct this model, first the number of possible views must be specified. For instance, to model the box in Figure 1, we represent it with only 4 different views: left, frontal, right and top. For each of these views, we proceed exactly as described in section 2.2, creating one appearance model for each of the four views.

The 3D model is simply a combined list of 3D points corresponding to the landmark points in each view. Since these landmark points are obtained for each of the views separately, they need to be transformed into a common object coordinated frame. This is done by applying the 3D to 3D absolute orientation algorithm [10].

3.2. Tracking algorithm

Given the above representation, we can now explain how to track a 3D rigid object in a sequence of images. The tracking algorithm can be divided in two major steps. The first step, *Selecting Appearance Models*, determines a list of appearance models that should be visible in the current image. It is this step of our new approach that permits tracking in the presence of self-occlusion. The second step, *Robust Multiple-AAM Matching*, uses the visible appearance models detected in step one to estimate the 3D location of the object. It is in this step that partial occlusion is handled -- by introducing robust estimation in the AAM matching algorithm (section 3.2.2).

3.2.1. Selecting Appearance Models

In this step, self-occlusions are detected using a simple backface removal algorithm [13]. The procedure starts by creating a list of 3D triangular meshes from the 3D model of the object. In fact, for efficiency, this list of triangular meshes can be pre-calculated during the model construction.
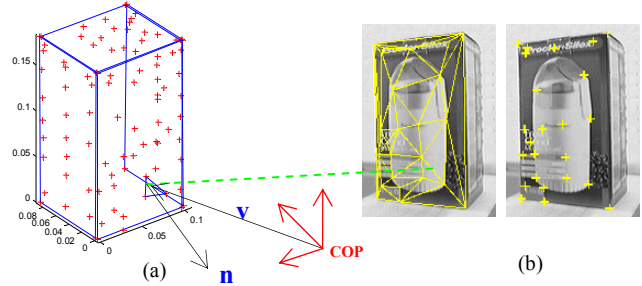

Figure 2. The appearance model of the box: (a) 3D triangular mesh from the 3D Model and (b) the corresponding 2D triangular meshes from the projections of (a)

Since, at each frame, we have the estimate of the object pose from the previous frame – given by $\mathbf{t}_{3D}$ – this information can be used to predict the current pose of the triangular meshes. That is, we apply the 3D rigid transformation given by $\mathbf{t}_{3D}$ to the coordinates of each vertex of a mesh and then determine the normal vector with respect to the camera coordinate frame. This procedure is possible because we assume that the object pose in two consecutive image frames does not change significantly. This assumption and the consequences when it fails to hold are discussed in section 4.

For each triangular mesh, the algorithm calculates the viewing vector $\mathbf{v}$ that points out from the origin of the camera coordinate frame (center of projection - COP) to the center of that mesh (Figure 2). Also, since we have calculated the normal vector ($\mathbf{n}$) of each triangular mesh, we can define the viewing angle Θ of the mesh using the dot-product between the two vectors above.

The visibility of the triangular mesh can be determined by testing the viewing angle Θ. If this angle is smaller 90 degrees, this means that the triangular mesh is not visible from the current viewing angle of the camera. Otherwise, the triangular mesh is visible. This visibility, however, increases with increasing values of the angle. To capture this property, we define the visibility of a triangular mesh as:

$$w = |\cos \Theta| \qquad (7)$$

For each appearance model, if at least one triangular mesh is visible according to the criterion above, that model is included in the list of selected appearance models. The above algorithm also creates another list with the visibility measures w. With these two lists, we can run the Robust Multi-AAM Matching algorithm, as explained next.

3.2.2. Robust Multiple AAM Matching

In order to handle partial occlusion, as mentioned already, we employ a robust estimation framework. The

underlying assumption of this framework is that the occluded area can cause large texture differences between the template and what is seen in the camera image. In that case, the texture difference corresponding to the occluded pixels can be considered as "oultiers" in the estimation process.

In the new proposed matching algorithm, we incorporate the standard M-estimation and the automatic scale computation [14] in our previous matching formulation. That is, in eq. (6), instead of using the L2-norm, we determine the parameters $\mathbf{p}=(\mathbf{b_g},\mathbf{t}_{3D})$ that minimize a similar expression using the robust norm $\rho()$. That is:

$$\mathbf{p} = \arg\min \sum_i w_i * \rho(r_i(\mathbf{p});\sigma) \qquad (8)$$

where r_i is texture residue at pixel i. The robust norm function $\rho()$ is defined by the Lorentzian function where the scale parameter σ is automatically estimated from the median value of the absolute residue for the initial parameter $\mathbf{p}^0$ (initialized with the result from the previous frame). That is, the robust norm function and the estimate of σ are automatically obtained by:

$$\rho(r_i;\sigma) = \log(1+\frac{r_i^2}{2\sigma^2}); \quad \sigma = 1.4826\, Med_i\,|r_i(\mathbf{p}^0)| \qquad (9)$$

Another change in eq. (8) with respect to eq. (6) is in the incorporation of a weight w_i, which is applied to each residue r_i. This weight is derived from the visibility measure in eq (7) and it represents the credibility of each pixel used in the estimation.

To perform the above minimization, we also had to modify the Gauss-Newton formulation to accommodate the M-estimation. Our modification to the Gauss-Newton formulation is along the lines of the iterative method presented in [14]. In this method, given the current estimate of $\mathbf{p}$, we determine the next estimate $\mathbf{p'}= \mathbf{p}+\Delta\mathbf{p}$ by calculating $\Delta\mathbf{p}$ using the gradient $\mathbf{G}$ and the first order estimate of the Hessian $\mathbf{H}$ of the objective function as in:

$$\Delta\mathbf{p}=-\mathbf{H}^{-1}(\mathbf{p})\,\mathbf{G}(\mathbf{p}) \qquad (10)$$

where the kth element of $\mathbf{G}$ and the (k^{th},l^{th}) elements of $\mathbf{H}$ can be written in terms of ρ and r_i as

$$G_k = \sum_i w_i \frac{\partial\rho}{\partial r_i}\frac{\partial r_i}{\partial p_k} \quad ; \quad H_{kl} = \sum_i \frac{w_i}{r_i}\frac{\partial\rho}{\partial r_i}\frac{\partial r_i}{\partial p_k}\frac{\partial r_i}{\partial p_l} \qquad (11)$$

Now that we can handle partial occlusion using robust estimation, we must combine this with the previous step where we select the multiple appearance models that are visible in the current image. It is the combination of both steps that will allow our algorithm to handle both partial and self occlusions.

Let's assume that the frontal and the left views of the object are visible in the current frame. There exists a separate appearance model for each view. The parameter vectors corresponding to each of these models in the image can be denoted $\mathbf{b_g}_{V_1}$ and $\mathbf{b_g}_{V_2}$. The matching algorithm needs to take into account both of these vectors at the same time. The matching algorithm needs to warp the texture from all visible triangular meshes in the current image to the reference templates for the two appearance models. The triangular meshes in the current image are obtained from the projection of all visible 3D triangular meshes determined in the previous step.

In general, to include all visible models in the estimation process, we dynamically combine the models as if they were one single appearance model. Then, we apply the robust estimation process to estimate the parameters that minimizes the residue over the combined model texture and the combined observed texture.

In other words, if $\{v_1, v_2, \ldots, v_k\}$ is the list of the visible appearance models selected by the first step of the algorithm, then the parameters being estimated are given by:

$$\mathbf{p} = (\,\mathbf{t}_{3D}\;\mathbf{b_g}_{V_1}\;\ldots\;\mathbf{b_g}_{V_k}\,) \qquad (12)$$

where $\mathbf{b_g}_{V_i}$ is the texture parameter vector corresponding to the $v_i{}^{th}$ appearance model. Also, at each iteration of the minimization process, given a current set of parameters, $\mathbf{p}$, we can obtain the combined model texture and the combined observed texture from, respectively:

$$\mathbf{g} = (\,\mathbf{g}_{V_1}{}^T\,|\,\ldots\,|\,\mathbf{g}_{V_k}{}^T\,)^T \quad \mathbf{g_o} = (\,\mathbf{go}_{V_1}{}^T\,|\,\ldots\,|\,\mathbf{go}_{V_k}{}^T\,)^T \qquad (13)$$

where $\mathbf{g}_{V_i}$ is the texture calculated from the $v_i{}^{th}$ appearance model (eq. (2)) with parameter $\mathbf{b_g}_{V_i}$ and $\mathbf{go}_{V_i}$ is the texture sampled from the image inside the projection of the meshes belonging to the appearance model v_i.

Finally, the residue vector, $\mathbf{r}(\mathbf{p})= \mathbf{g} - \mathbf{g_o}$, can be evaluated and used in the minimization process stated in equations. (8) through (11).
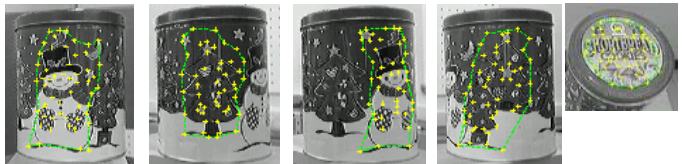

Figure 3. The "can" object at 5 different views with their annotated landmark points

4. Results

In order to evaluate our method, we collected experimental data using a pair of stereo cameras mounted on the end-effector of an industrial robot. The robot moved on several specified paths around a stationary target object. A sequence of images (640x480 pixels) was acquired and the end-effector's position was recorded for each image taken. Also, for all experiments reported here, we used two different target objects: a box and a can. The box was modelled using 4 different appearance models (frontal, top, left and right) as shown in Figure 1, while we used 5 different appearance models for the can, as shown in Figure 3. For each object, we acquired two different image sequences: one for training (approximately 10 images per

model) and another for testing (over 500 images for each object).

Finally, since the image sequences recorded in the manner described above presented only self-occlusion, in order to also create partial occlusion we overlapped a portion of the images of the target object with a small *patch* obtained from another independent image (Figure 4). This partial occlusion obscures between 20% and 40% of the total area of the target object

The qualitative result of our method can be observed in Figure 4. As that figure shows, the projected object border computed from the estimated pose of the object – represented by the yellow dotted lines superimposed onto the image – closely matches the actual border of the object (box), or the area defined by the exterior landmark points (can). The matching was good for all poses of the two objects, regardless of the presence of partial occlusion.
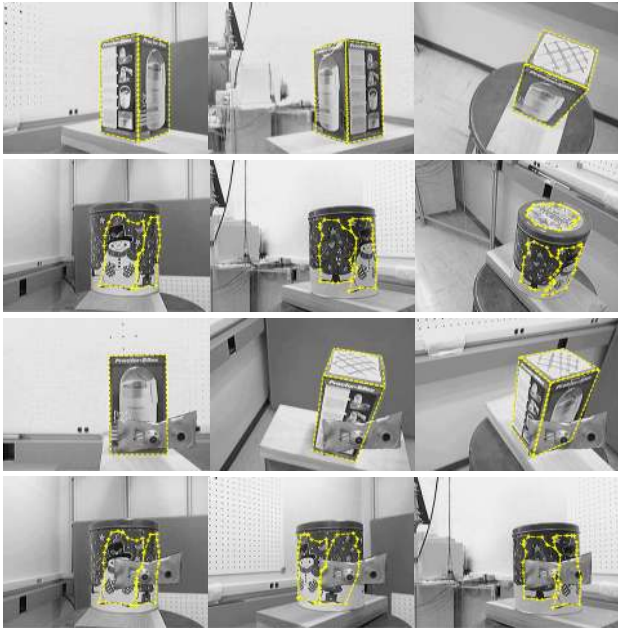

Figure 4. Some tracking results: with and without partial occlusion

For a quantitative assessment, we determined the accuracy of our method by comparing the estimated pose of the end-effector with the same pose obtained using the robot kinematics (ground truth). The pose of the end-effector is expressed using a homogeneous transformation by $^{ei}\mathbf{H}_{e_0}$ and it represents the end-effector's position "ei" with respect to some previously specified reference end-effector position "e$_0$". Note that while the target object is stationary with respect to the end-effector, the motion represented by $^{ei}\mathbf{H}_{e_0}$ is relative. That implies that even though it is the end-effector that is moving with respect to the object, it is the same as assuming that the end-effector is stationary while the object is moving.

In Figure 5, we depict the ground truth – that is, the relative motion of the box – in terms of the translation and the rotation components. Also in the same figure, we plot the estimated motion obtained from our method. The figure also shows the estimated motion in the presence of partial occlusion. Note that, due to space limitations, we plotted only the curve for the experiment using the box. Similar results were obtained for the can and are summarized in Table 1.

As it can be seen in Figure 5, the estimated poses are almost the same as the ground truth. This fact can be better appreciated in Table 1, where the statistics of the error between estimated and actual pose are summarized for each of the 6 degrees of freedom. In that table, we observe that the average error in any translational component is always smaller than 1cm with a very small standard deviation, usually less than 0.5cm. In terms of the rotational error, on the average the error is less than one degree.
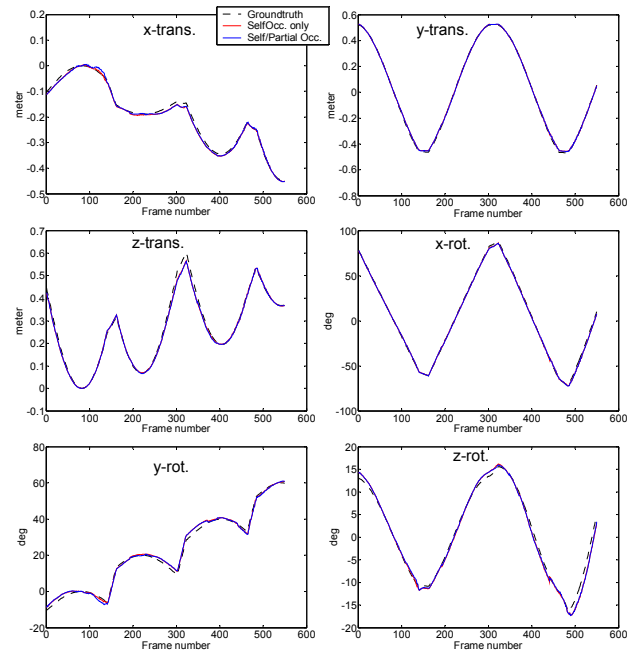

Figure 5. The motion paths for the "box" in experiment. Each plot shows the groundtruth and the calculated path using our method in the presence of self-occlusion only and both self and partial occlusion.

The relatively large values for the maximum error for both the translational and rotational components are due to a few "bad" samples, as it can be observed in Figure 5 (around frame 300) and as can also be inferred from the small standard deviations.

As we explained in section 3.2, we made an assumption that the object pose in two consecutive frames does not change significantly. This assumption affects the selection of the appearance models as described in section 3.2.1 as well as the initialization of the parameter $\mathbf{p}^0$ in section 3.2.2. The latter is critical since a bad initialization of $\mathbf{p}$ may cause the matching algorithm to not converge.

In order to check the consequences of the failure to satisfy this assumption, we tested our method for several velocities of the target object. In Figure 6, we show the error in pose estimation for the box moving from 1 to 7 cm between two consecutive frames. As can be observed, the

method works well – no change in pose error – for any amount of motion of the box and without occlusions. However, in the presence of occlusion, the error grows rapidly. We believe this growth is due mainly to the bad initialization of the parameter **p**, more specifically, $\mathbf{t}_{3D}$. If the tracking algorithm starts with a bad value for $\mathbf{t}_{3D}$, the texture samples will be taken from an image region far away from the expected one. This adds to the fact that the object is being occluded and it causes an increasingly large number of outliers in the robust estimation, which is known to fail for more than 50% of outliers.

Table 1.

| Statistics of error			x-trans (mm)	y-trans (mm)	z-trans (mm)	x-rot (deg)	y-rot (deg)	z-rot (deg)
B O X	Self-occ. Only	Avg	7.0	6.2	9.3	1.16	1.08	0.68
		Std	4.7	4.9	10.6	0.87	0.71	0.63
		Max	18.1	22.1	41.1	4.49	2.57	2.61
	Self/ Partial occ.	Avg	7.1	6.5	9.4	1.25	1.18	0.67
		Std	5.0	4.5	11.2	0.88	0.79	0.62
		Max	18.1	21.4	44.7	4.27	2.73	2.61
C A N	Self-occ. Only	Avg	4.8	5.2	7.5	0.76	0.48	0.63
		Std	3.6	3.2	5.5	0.56	0.32	0.39
		Max	16.9	18.6	33.6	2.98	1.42	1.60
	Self/ Partial occ.	Avg	4.5	5.1	7.4	0.79	0.45	0.65
		Std	3.4	3.5	5.5	0.60	0.34	0.42
		Max	17.0	18.9	33.4	3.00	1.52	1.67
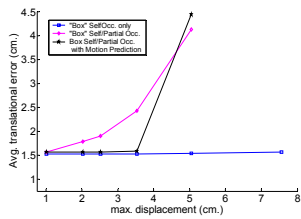
Figure 6. The graph shows the avg. translational error for different speeds of the object.

We tested this hypothesis by implementing a very simple linear-motion prediction scheme to initialize **p** for the next frame. When we use this prediction, the error improves again for motions up to 3.5 cm between consecutive frames. But it fails again for motions of about 5 cm between consecutive frames because larger motions also involve changes in the direction of the end-effector motion --- especially because such changes are not incorporated in the simple prediction scheme used.

5. Conclusion and Future Work

We introduced a new method to track rigid objects using a modified version of AAM. The method allows for partial and self occlusions of the object. We showed results for wide changes in the position and orientation of the object. For the experiments shown, tracking was carried out with high precision over a range of 1 meter in translation and 140 degrees in rotation.

The proposed method suffers from one drawback that is common to many trackers --- the tracker works only when the frame-to-frame variations in the object pose are small.

We also showed that this problem could be partially alleviated by incorporating simple motion prediction in the tracking process. But, for obvious reasons, further works needs to be carried out on how best to carry out motion prediction in our context. Another major drawback of the method is that the initialization of the pose parameters must be carried out manually in the first frame of the video sequence. Practically all trackers today that analyze targets at a level higher than merely considering them to be blobs suffer from this limitation. This initialization issue also needs to be addressed if the type of trackers presented here are to have any real-world applications.

Since our main goal is to use this method for visual servoing, we must improve the speed of the matching algorithm so it can run in real time. The most time consuming step of the algorithm is for computing the Hessian matrix and the gradient vector (in eq (10) - (11)). We are currently working on the development of a real-time version of the algorithm based on the inverse compositional approach [15].

6. References

[1] D.Gennery, "Visual tracking of known three-dimensional objects," IJCV, 7:243-270, 1992.

[2] D.Koller, K.Daniilidis, and H.H. Nagel, "Model-based object tracking in monocular image sequences of road traffic scenes," IJCV, 10:257-281, 1993.

[3] V. Blanz and T.Vetter, "Face Recognition Based on Fitting a 3D Morphable Model," IEEE Trans. PAMI, Vol.25, pp. 1063-1074, Sep 2003.

[4] S. Romdhani, A. Psarrou, and S. Gong. "A Generic Face Appearance Model of Shape and Texture under very large Pose Variations from Profile to Profile Views," Proceedings of the 15th ICPR 2000, Vol I: 1060-1063.

[5] T.F. Cootes, G.J. Edwards, and C.J. Taylor, "Active Appearance Model," IEEE Trans. PAMI, Vol.23 (6), pp. 681-685, June 2001.

[6] T.F. Cootes, K.Walker, and C.J. Taylor, "View-based Active Appearance Models," Image and Vision Computing, Vol 20, pp. 657-664, 2002.

[7] Y. Wu, T. Yu and G. Hua, "Tracking appearance with occlusions," IEEE Conf. CVPR. June 2003, vol. I, pp.789-795.

[8] A. Pentland, B. Moghaddam, and T. Starner, "View-based and Modular eigenspaces for face," IEEE Conf. CVPR June 1994, pp.84-91.

[9] J. Ahlberg, "Using the active appearance algorithm for face and facial feature tracking," in IEEE ICCV workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems, 2001, pp. 68-72.

[10] B.K.P. Horn, "Closed-form solution of absolute orientation using unit quarternions," Journal of Optical Soc. Am. , Vol. 4, pp. 629-642, 1987.

[11] P. Mittrapiyanuruk, G. N. DeSouza, A. C. Kak, "Calculating the 3D-pose of Rigid-Objects using Active Appearance Models," Proceedings of IEEE ICRA 2004

[12] J. Xiao, S. Baker, I. Matthews, and T. Kanade, "Real-Time Combined 2D+3D Active Appearance Models" Proceedings of the IEEE Conf. on CVPR, June, 2004.

[13] C.Pokorny, Computer Graphics: An object-orientation Approach to the Art and Science, Franklin, Beedle & Associates, c1994.

[14] H.S. Sawhney and S. Ayer, "Compact Representations of Videos Through Dominant and Multiple Motion Estimation," IEEE Trans. PAMI Vol.8 (8) Aug. 1996.

[15] S. Baker and I. Matthews, "Lucas-Kanade 20 Years On: A Unifying Framework," IJCV, Vol. 56 (3), March, 2004, pp. 221 - 255.

[16] R. Gross, I. Matthews, and S. Baker, "Construction and Fitting Active Appearance Models with occlusion," in the Proceedings of the IEEE Workshop on Face Processing in Video, 2004