This paper appears in: IEEE International Conference in Robotics and Automation, 2004

Calculating the 3D-Pose of Rigid-Objects Using Active Appearance Models

Pradit Mittrapiyanuruk, Guilherme N. DeSouza, Avinash C. Kak

{mitrapiy,gdesouza,kak}@ecn.purdue.edu

Robot Vision Laboratory, School of Electrical and Computer Engineering, Purdue University

Abstract— **This paper presents two different algorithms for object tracking and pose estimation. Both methods are based on an appearance model technique called Active Appearance Model (AAM). The key idea of the first method is to utilize two instances of the AAM to track landmark points in a stereo pair of images and perform 3D reconstruction of the landmarks followed by 3D-pose estimation. The second method, the AAM matching algorithm is an extension of the original AAM that incorporates the full 6 DOF pose parameters as part of the minimization parameters. This extension allows for the estimation of the 3D pose of any object, without any restriction on its geometry. We compare both algorithms with a previously developed algorithm using a geometric-based approach [14]. The results show that the accuracy in pose estimation of our new appearance-based methods is better than using the geometric-based approach. Moreover, since appearance-based methods do not require *customized* feature extractions, the new methods present a more flexible alternative, especially in situations where extracting features is not simple due to cluttered background, complex and irregular features, etc.**

I. Introduction

Determining the position and orientation (the 6 DOF pose) of an object plays an important role in many research areas such as robotics, computer graphics, computer vision, etc. In robotics applications, the exact pose of an object is crucial in order to control the robot end-effector to perform any specific task on that object.

A classical approach for determining the pose of an object relies on the object geometrical features and is usually referred to as feature-based pose estimation [6, 7, 3]. The basic strategy in feature-based approaches is to match observed features (scene) and expected features (model). This match can be performed either by projecting the model features onto the image plane [6, 7] or by calculating the 3D coordinates of the observed features and determining the transformation that brings observed and expected features into registration [3]. Feature-based methods are usually very accurate because once a first match between observed features and expected features is obtained, the next matches can be constrained by the geometry of the object, and every match thereafter improves the accuracy by reducing the uncertainty in the matchings. However, due to the complexity of the object and the background, extracting and matching features can be daunting, and as an alternative, there are the appearance-based methods.

In appearance-based approaches, a set of training images is used to calculate a covariance space (eigenspace). Future observed images are matched to the training data using many distance measurements. The matching between the observed image and trained appearances is key in the strategy used for either pose estimation or object recognition. In [4], it was proposed an appearance-based pose estimation method for robot positioning in 2 DOF. Later, [5] extended this idea to the full 6-DOF pose of an object but with limited accuracy.

The main advantage of appearance-based methods is that no features – simple or complex – need to be extracted, which in feature-based methods needs to be done on a per object basis. The main drawback of the traditional appearance-based methods is in the initial object segmentation and subsequent 2D alignment (translation) and scaling of the observed object in the image so that it can be compared to the model. Some authors [4, 5] have proposed a simple normalization and centering of the image samples before the appearance matching can be performed. However, there is no parameterization of the translation or scaling in these methods, and therefore this pre-processing can affect the accuracy of the pose estimation. Several other works [10, 11] resort to some type of 2D image warping, which is iteratively applied during matching. For obvious reasons, these methods are also not applicable to 6 DOF pose estimation.

In [12], a more relevant method is introduced to solve the pose estimation of human heads in 6 DOF. In this case, an image warping is derived from the projection of the 3D model of the human head – represented by a standard-size cylinder. While this method is reasonable for human head tracking, it imposes various constraints on the shape of the object, making it hard to be applied to more generic objects – which is the main focus of this paper.

Finally, there are the works derived from the Active Appearance Models paradigm (AAM) [8, 9]. In AAM, the problem of image warping is handled by defining a set of sparse landmark points around the object. Besides controlling the warping, these points play a key role in representing the object appearance – defined by the shape around the landmark points and the texture inside the shape. Therefore, the arbitrary choice of landmark points allows for the modeling of various shapes of objects.

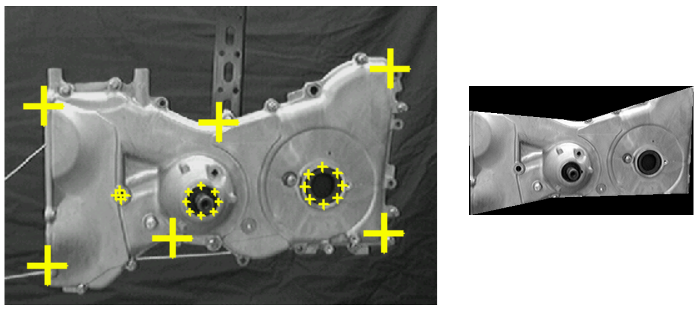
Some work has also been done using AAM to track rigid objects [17] in the image plane. However, as far as we know, AAM has never been applied to vision-guided robotics (e.g. visual servoing), where the pose of the object must be determined in all 6 DOF, and accuracy is a major requirement.

In this paper, we propose two different algorithms using AAM to tackle the problem of object pose tracking in 6 DOF. The first algorithm uses stereo vision and two appearance models: one for the left and one for right images. The second algorithm uses only one camera and one appearance model.

The rest of this paper is organized as follows: in Section II we present a brief review of Active Appearance Model, followed by a detailed explanation in Section III of our AAM-based methods for tracking and pose estimation. Finally in Section IV we will discuss the results.

II. Review of Active Appearance Models (AAM)

In this section, a brief review of AAM is presented[1]. In simple terms, the main idea in AAM is to learn a parametric function that governs the possible appearances of an object. That is, if we can learn how to change a set of parameters in order to synthesize any desired appearance of the object, then we can find a perfect match between observed appearance (current image) and expected appearance (model). This information can then be used to recognize the object and its pose.



(a) (b)

Figure 1. (a) The object image with the landmark points (cross mark), (b) The image region covered by the landmark points

A. Modeling the Appearance

In AAM, the appearance of an object is defined by its shape and texture. The shape of the object is a set of 2D landmark points of the object image. That is, the shape is represented by a vector $\mathbf{x} = [x_1\ y_1\ ..\ x_n\ y_n]^T$, where the element $(x_i\ y_i)$ is the x-y coordinates of the ith landmark point. Similarly, the texture of the object is defined by the set of intensity values of the pixels lying inside the shape. That is, the texture of the object is a vector $\mathbf{g} = [g_1\ g_2\ ..\ g_m]^T$, where g_i is the intensity value of the ith pixel. For example, Figure 1a depicts our experimental target object, and Figure 1b shows the corresponding texture pixels for the landmark points (shape) indicated in Fig. 1a.

[1] Because of the space limitation, we will discuss only the main concept that is required to understand our object pose tracking methods. For a more comprehensive explanation, we refer the reader to [9].

From this point on, AAM models the appearance of an object in a very similar way as in other appearance-based methods. In other words, a PCA analysis of the training samples is performed to obtain: the mean shape vector ($\bar{x}$); the basis (eigenvectors) of the space defined by the covariance of the shapes $\mathbf{P}_s$; the mean texture vector ($\bar{g}$); and the basis of the covariance space of the textures $\mathbf{P}_g$. Using these vectors and matrices, the appearance of an object can be described by:

$$x = \bar{x} + P_s b_s \tag{1}$$
$$g = \bar{g} + P_g b_g \tag{2}$$

where $\mathbf{b}_s$, $\mathbf{b}_g$ are respectively the set of shape and texture parameters that characterize a specific shape and texture sample.

After a few mathematical manipulations, a single set of parameters $\mathbf{c}$ can be derived for both shape and texture. The combined model allows an object to be described by:

$$x = \bar{x} + Q_s c \tag{3}$$
$$g = \bar{g} + Q_g c \tag{4}$$

Note that the coordinates of the shape vector $\mathbf{x} = [x_1\ y_1\ ..x_n\ y_n]^T$ are expressed in the model coordinated frame. In other words, a 2D rigid transformation $\mathbf{T}(.)$ with translation, orientation, and scaling parameters $\mathbf{t} = [tx,\ ty,\ s,\ \Theta]^T$ is applied to transform the points $\mathbf{x}$ to/from the pixel-coordinate points $\mathbf{X}$. That same transformation allows the image warping that aligns the observed texture with the texture model.

B. AAM Matching

For the matching phase, AAM defines a parameter vector

$$\mathbf{p} = (\mathbf{c} \mid \mathbf{t})^T \tag{5}$$

where c is the set of appearance parameters and t is the 2D rigid transformation parameters;

a texture-difference vector function

$$\mathbf{r}(\mathbf{p}) = g_s(\mathbf{p}) - g_m(\mathbf{p}) \tag{6}$$

where $\mathbf{g}_s(\mathbf{p})$ is the observed texture vector and $\mathbf{g}_m(\mathbf{p})$ is the model texture vector;

and finally an error function

$$e(\mathbf{p}) = \mathbf{r}(\mathbf{p})^T\,\mathbf{r}(\mathbf{p}) = \|\,\mathbf{r}(\mathbf{p})\,\|^2 \tag{7}$$

The matching algorithm consists of an iterative minimization process to find the set of new parameters $\mathbf{p}$ that minimizes e(.). That is,

$$\Delta\mathbf{p}^* = \operatorname{argmin}_{\Delta\mathbf{p}}\, e(\mathbf{p}+\Delta\mathbf{p}) \tag{8}$$

which after applying the Taylor series and solving for Δp^*, leads to:

$$\Delta\mathbf{p}^* = -\mathbf{R}(\mathbf{p})^*\mathbf{r}(\mathbf{p}) \tag{9}$$

where $\mathbf{R}(\mathbf{p})$ is the pseudo-inverse of the gradient (Jacobian) matrix $\frac{\partial}{\partial p} r(p)$

The AAM matching algorithm is quite simple and it states that at each iteration, a set of parameters $\mathbf{p}$ is used to generate the model shape vector $\mathbf{x}$ and the texture vector $\mathbf{g}_m(\mathbf{p})$ of the object according to Eq. 3 and 4. The model shape is then projected on the image plane and the current texture $\mathbf{g}_s(\mathbf{p})$ is sampled from the image. Next, the two textures are compared, that is, the difference between model texture and observed texture is calculated according to Eq. 6 and the value of the error function $e(\mathbf{p})$ is determined by Eq. 7. Finally, a new set of parameters $\mathbf{p} = \mathbf{p} + \Delta\mathbf{p}^*$ can be calculated from Eq. 9. The process above repeats until the error function $e(\mathbf{p})$ converges to a small value.

As it will be explained in Section 3, to speed up the process, the original AAM algorithm assumes a fixed value for the matrix $\mathbf{R}(\mathbf{p})$ in Eq. 9. However, in our method we observed that this approximation causes a large error and instead we decided that a new value of $\mathbf{R}(\mathbf{p})$ must be calculated at each iteration.

III. POSE ESTIMATION METHODS USING AAM

As mentioned earlier and explained in detail in [1, 2], our main application for this work is to servo the end-effector with respect to an object (visual servoing). In order to do that, we must first find the pose of the object with respect to the end-effector – given by the homogenous transformation[2] ${}^{e}\mathbf{H}_{O}$. Since a pair of stereo cameras is mounted onto the end-effector, this pose can be decomposed into:

$$ {}^{e}\mathbf{H}_{O} = {}^{e}\mathbf{H}_{C} * {}^{c}\mathbf{H}_{O} \tag{10} $$

where ${}^{e}\mathbf{H}_{C}$ is a constant HTM that is calculated during the "hand-eye" calibration [1] and c represents either the left or right camera, while ${}^{c}\mathbf{H}_{O}$ is the actual pose of the object, which we want to estimate.

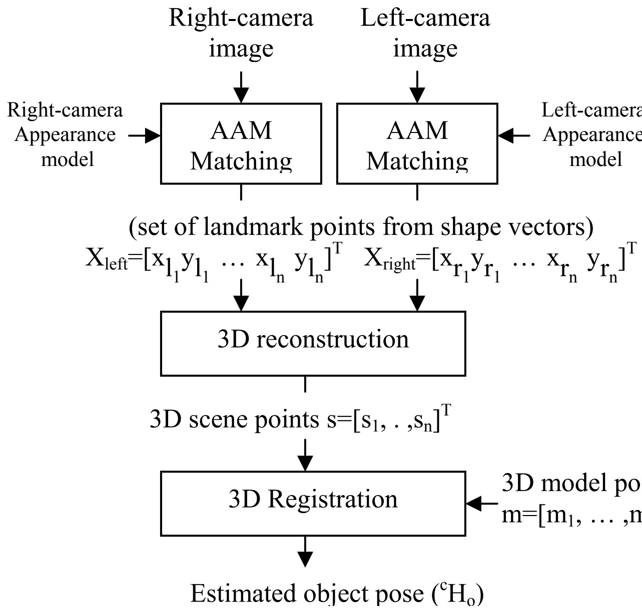
In the next two subsections we will describe two methods for pose estimation using AAM. Both our methods start with an off-line training phase to create simultaneously the appearance models and a 3D representation of the object. This 3D representation consists of a list of 3D points of the object with respect to its own coordinate frame. In other words, we developed a simple tool that allows a human to choose the landmark points that define the shape vector from a set of training stereo images. As the human selects the landmark points, the corresponding list of 3D coordinates of the landmarks is also generated. The tool guarantees that a series of constraints, e.g. epipolar constraint, is imposed on top of the human selections. As it will be explained later, these 3D model points are necessary to calculate the actual pose of the object in space.

[2] In this paper, we assume the Denavitt-Hartenberg notation for HTMs

A. *The first method: Stereo AAM*

The key idea in our first method, Stereo AAM, is to apply the AAM algorithm to do image tracking in both left and right images, and use the stereo pair of tracked landmark points to calculate the 3D pose of the object. Our complete Stereo AAM algorithm can be divided in three steps (Figure. 2):

- Stereo AAM matching;
- 3D Reconstruction; and
- 3D Registration.



Right-camera image
Left-camera image
Right-camera Appearance model
AAM Matching
AAM Matching
Left-camera Appearance model
(set of landmark points from shape vectors)
$X_{left}=[x_{l_1} y_{l_1} \dots x_{l_n} y_{l_n}]^T$
$X_{right}=[x_{r_1} y_{r_1} \dots x_{r_n} y_{r_n}]^T$
3D reconstruction
3D scene points $s=[s_1, . . ,s_n]^T$
3D Registration
3D model points $m=[m_1, \dots ,m_n]^T$
Estimated object pose (${}^{c}H_{o}$)

Figure 2. Block diagram of our first method

In the first step, two independent AAM matching algorithms are applied: one for the right and one for the left image. To apply the AAM matching, we rely on the above-mentioned appearance models that were built during the training phase. Moreover, since each landmark point that was annotated on the training images has a unique correspondence in the generated 3D model, a simple ordering of the landmark points in the left and right images can be established. This ordering will be necessary during the 3D Reconstruction step of our algorithm.

A typical problem in AAM is to obtain an initial guess for the parameters used in the matching algorithm. In our visual servoing system, another subsystem called Coarse Control provides this initial guess for the first frame [2]. Thereafter, the Stereo AAM method is applied to all subsequent frames and the initial guess for each frame is derived from the estimated parameters calculated for the previous frames. This not only solves the initialization problem, but also helps the AAM matching converge quickly.

The second step of the algorithm is the 3D Reconstruction. As we mentioned above, the Stereo AAM Matching provides a simple order for the landmarks in the left and right images. After the match is obtained, we can calculate the 3D coordinates of the landmark points trivially using the stereo correspondence provided by the order of the landmarks. That is, from the right shape vector $\mathbf{X}_{right}=[x_{r_1} y_{r_1} \ldots x_{r_n} y_{r_n}]^T$ and the left shape vector $\mathbf{X}_{left}=[x_{l_1} y_{l_1} \ldots x_{l_n} y_{l_n}]^T$ we reconstruct the 3D scene points s_i from the pairs $((x_{l_i}, y_{l_i}), (x_{r_i}, y_{r_i}))$ using a simple linear 3D reconstruction method [15].

For the final step of our first method, 3D Registration, we used a 3D absolute orientation algorithm [13]. Basically the goal of the absolute orientation algorithm is to determine the transformation that relates two different sets of 3D points. That is, given the 3D model points in the object coordinate frame and the 3D scene points in the camera coordinate frame the absolute orientation algorithm calculates the object pose given by ${}^{c}\mathbf{H}_o$ – the homogeneous transformation that relates the camera coordinate frame and the object coordinate frame.

In other words, given the 3D model points $\mathbf{m}_i$ and the 3D scene points $\mathbf{p}_i$, we can find the rotation $\mathbf{R}$ and translation $\mathbf{T}$ that define ${}^{c}\mathbf{H}_o$ and satisfy:

$$s_i = \mathbf{R}*\mathbf{m}_i + \mathbf{T} + \mathbf{e}_i \tag{11}$$

where e_i is the summation of all expected errors: the 2D landmark localization obtained by the AAM matching; the accuracy of 3D-reconstruction; pose calculation; etc.

The rotation and translation above are obtained by solving a minimization problem defined by:

$$R_{opt}, T_{opt} = \arg\min_{R,t} \sum_{i=1}^{n} \| s_i - R*m_i - T \|^2 \tag{12}$$

For that, we used a closed-form solution method based on unit-quartenions [13].

B. The second method: AAM using a 3D-pose parametrization

Our second method[3] is a generalization of the AAM algorithm. As we mentioned in Section II, the original AAM uses a parameter vector p:

$$\mathbf{p} = (\mathbf{c} \mid \mathbf{t})^T$$

where $\mathbf{t} = [t_x \; t_y \; s \; \Theta]$ is a vector describing the 2D transformation: translation, rotation, and scaling. In our method, we extended this definition to include all 6 DOF translation and rotation of the object. Therefore, the new t becomes:

$$\mathbf{t} = (t_x \, t_y \, t_z \, r_x \, r_y \, r_z)$$

Another modification with respect to the original AAM is in the combined texture and shape parameter vector $\mathbf{c}$. In the original AAM, this parameter vector is used to determine both the shape and the texture vectors. That is, the shape defined by the landmark points $\mathbf{x}$ and the texture (intensities of points that lie inside the shape) $\mathbf{g}$. In our approach, the 3D translation and rotation above – given by $\mathbf{t}$ – together with the intrinsic parameter of the camera provide the shape vector. This is done by projecting the 3D model points onto the image plane. Once the shape vector is determined, the observed texture points, $\mathbf{g}_S$, can be sampled from the image. Consequently, since we do not need to determine the landmark points and the texture points from equations 3 and 4 any longer, and since we only need to determine the model texture points from Eq. 2, we only use b_g in the parameter vector p. That is, we use $\mathbf{p}=(\mathbf{t} \mid \mathbf{b}_S)$ instead of $p=(\mathbf{t} \mid \mathbf{c})$.

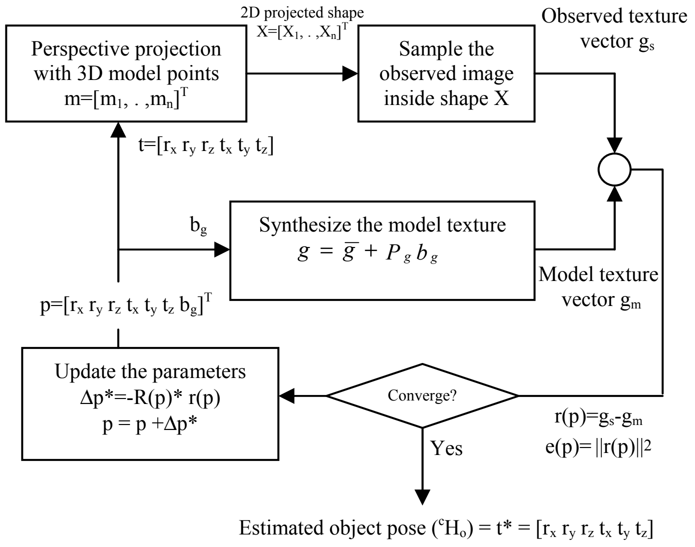

2D projected shape
X=[X1, . ,Xn]T
Perspective projection with 3D model points m=[m1, . ,mn]T
Sample the observed image inside shape X
Observed texture vector gs
t=[rx ry rz tx ty tz]
bg
Synthesize the model texture g = ḡ + Pg bg
Model texture vector gm
p=[rx ry rz tx ty tz bg]T
Update the parameters Δp*=-R(p)* r(p) p = p +Δp*
Converge?
Yes
r(p)=gs-gm
e(p)= ||r(p)||2
Estimated object pose (cHo) = t* = [rx ry rz tx ty tz]

Figure 3. Block diagram of our second method

In order to explain our algorithm step by step we refer the reader to Figure 3. From the top left box, the process begins by providing a set of initial values for p, given by:

$$\mathbf{p} = [t_x \, t_y \, t_z \, r_x \, r_y \, r_z \, \mathbf{b}_g]$$

These rotational and translational parameters are used to project the model points $\boldsymbol{m_i}$ onto the image plane to obtain the set of landmark points $\mathbf{X}_i$ – the shape vector in image coordinates. With the shape vector, we can sample the texture points inside the observed image and calculate $\mathbf{g}_S$, just as it is done in the original AAM algorithm. Next a texture model $\mathbf{g}_m$ is generated using the current $\mathbf{b}_g$ in Eq. 2, and the texture error is calculated using Eq. 6. Finally, a new set of parameters $\mathbf{p}$ can be obtained using Eq.9. The process above repeats until the difference between model texture and observed texture, measured by e($\mathbf{p}$) in Eq. 7, converge to a small value.

An important difference with respect to our previous method – Stereo AAM – is that this method does not require the 3D-reconstruction phase. Therefore, we can use only one set of images, that is, from a single camera. Nevertheless, we still use both pair of images to create the 3D model as explained in the beginning of Section III.

[3] A similar method using AAM and a 3D-pose parameterization for t is employed in [18]. However, in their approach the texture parameter b_g is not included in the final parameterization p. Instead, they approximate the model texture using the current observed texture vector, which from our experience can produce errors in the AAM matching.

IV. Results and Discussion

In order to evaluate our methods[4], we set up a workspace to simulate an automotive assembly cell. The workspace consists of a target object (the engine cover shown in Figure 1.), a Kawasaki UX-120 robot, and two stereo cameras mounted onto the robot end-effector. The end-effector can move freely around the target object, which remains still while the cameras grab image sequences from various viewpoints. This way we can determine the exact relative pose of the object, which will be used as the ground truth, as we will explain shortly.

In order to obtain the experimental data, we moved the robot end-effector on five arbitrary paths around the target object. While the end-effector was moving we acquired the stereo image sequence as well as the end-effector position at the time when each stereo image was acquired.

We stored two different stereo sequences: one for training and one for testing. For the training sequence, we selected 26 stereo images from different viewpoints, and on each image we selected 26 landmark points as shown in Figure 1a. After running a PCA analysis of the training images, we constructed both the left and right appearance models, with each model reaching a dimensionality of 10 – which represents 98% of the variation determined from the PCA analysis. In other words, the size of parameter vector c was chosen to be 10.

Finally we tested both our methods on 217 frames of the testing data. Since our second method, the 3D-pose parameterization, only requires monocular camera, we used only the left-camera images and the left appearance model in this algorithm.

As we mentioned earlier, we compared both algorithms to our previously developed algorithm using geometric features (circular shapes). This algorithm was shown [14] to be sufficiently accurate for our applications and it seemed to be a good reference to compare our new algorithms.

In order to measure the accuracy of the algorithms, we calculated the ${}^{ei}\mathbf{H}_{e_0}$ transformations. This homogenous transformation matrices represent the end-effector position "ei" with respect to the reference end-effector position "e_0" – where e_0 is an arbitrarily provided pose of the end-effector and "i" is the instant when an image frame was sampled. Since it is the end-effector that moves while the object remains stationary, we can obtain the ground truth for each frame by reading the robot controller and obtaining the transformation ${}^{ei}\mathbf{H}_b$ (direct kinematics). The ground truth is then easily obtained by:

$$({}^{ei}\mathbf{H}_{e_0})_{ref} = {}^{ei}\mathbf{H}_b * ({}^{e0}\mathbf{H}_b)^{-1} \tag{13}$$

Meanwhile, using both our methods, we estimated the same HTM by calculating the pose of the object with respect to the end-effector ${}^{e}\mathrm{H}_o$ at each frame 'i', that is:

$$({}^{ei}\mathbf{H}_{e_0})_{est} = ({}^{ei}\mathbf{H}_o) * ({}^{e0}\mathbf{H}_o)^{-1} \tag{14}$$

[4] Our methods were implemented using the AAM-API toolkit [16].

Finally we calculated the accuracy of the algorithms by comparing $({}^{ei}\mathbf{H}_{e_0})_{ref}$ – derived from the groundtruth – and $({}^{ei}\mathbf{H}_{e_0})_{est}$, – derived from the algorithms.

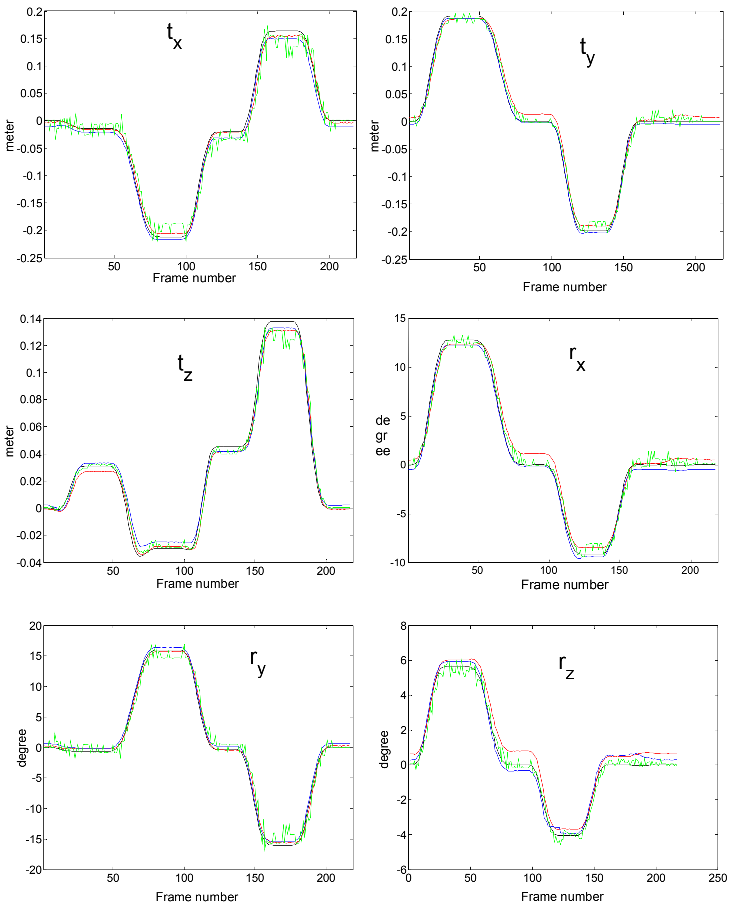


Figure 4. Groundtruth pose (black) and estimated pose of 1st method (red), 2nd method (blue) and geometrical-based method [14] (green).

In Figure 4, we show the accuracy measured in the experiments by plotting ${}^{ei}\mathbf{H}_{e_0}$ in terms of t_x, t_y, t_z, r_x, r_y, and r_z. The figure also shows the results obtained from the ground truth, from the two new methods, and from the geometric-based method.

The statistics of the error over the whole testing sequence are shown in Table 1. As it can be observed from the table, the average translational error (t_x, t_y and t_z) for both our methods was less than 10mm. While for the geometric-based method [14] it was over 14mm. Similarly, the maximum error for each of these methods was 27.7mm, 20.6mm, and 52.3mm for the first, second, and the geometric-based methods respectively, and the standard deviations: 0.0057, 0.0036, and 0.0112, respectively.

In terms of rotational error, both our methods also proved to be superior to our previous geometric-based method, with the second method showing an even better result than the first one – maximum rotational error in the first method was 1.77 degrees, while in the second method it was only 0.73 degrees.

From the results, we can conclude that the accuracy in pose estimation of our both methods is quite similar, with the second method being slightly better than the first one. Clearly our new methods outperform the geometric-based method, especially if we consider the maximum errors. Furthermore, we found that the stability of the estimation in both new methods is better than the one in [14]. That fact can be observed in the graphs in Figure 4 in the high frequency noise on top of the curve for the geometric-based method. This noisy estimation in the geometric-based approach can be harmful in our servoing system because it may cause the end-effector to move jerkily.

TABLE 1.

Error	Stat.	1st method	2nd method	Geometric method [14]
t_x (m.)	Mean	0.0059	0.0098	0.0147
	Std	0.0055	0.0036	0.0112
	Max	0.0277	0.0206	0.0523
t_y (m.)	Mean	0.0097	0.0041	0.0079
	Std	0.0057	0.0019	0.0064
	Max	0.0249	0.0095	0.0290
t_z (m.)	Mean	0.0039	0.0042	0.0042
	Std	0.0029	0.0027	0.0045
	Max	0.0133	0.0143	0.0230
r_x (deg.)	Mean	0.6721	0.3857	0.4819
	Std	0.4048	0.1412	0.4002
	Max	1.5073	0.5638	1.6300
r_y (deg.)	Mean	0.4395	0.5507	0.9510
	Std	0.3570	0.0777	0.8083
	Max	1.7666	0.7285	3.7862
r_z (deg.)	Mean	0.5273	0.3246	0.2286
	Std	0.2564	0.1329	0.1808
	Max	1.0338	0.6639	0.9958

V. CONCLUSION

We presented two methods for tracking and pose estimation of generic objects using Active Appearance Models. Feature-based methods, which impose geometric constraints to calculating the pose of the object, tend to be very accurate, and therefore very useful for robotic applications. However, when we compare both our appearance-based methods to a third method based on the geometric features, both methods proved to be a lot better. The error in pose estimation using our second method with 3D parameterization produced a translational error of less than 10mm and a rotational error of little over one half of a degree.

Moreover, geometric-based methods require *customized* algorithms that usually need extensive adaptations when applied for a different object. On the other hand, the creation of appearance models for different objects is a mechanical task that can be easily performed. Another major advantage of appearance-based methods over feature-based methods is in situations where extracting features is not simply done due to cluttered background, complex and irregular features, etc.

On the downside, like any other appearance-based method, our methods cannot be applied when large variations of the object appearance are observed; for example, when the object is observed from a very different angle, or in the presence of occlusions, etc. In these cases, the learnt view of the object may be well hidden for a good match with the model to take place. We are currently developing a strategy that consists of learning different appearance models of the same object when observed from different angles. This multiple AAMs can then be tracked simultaneously while the object is observed from radically different views in the image sequences.

ACKNOWLEDGMENT

The authors would like to thank Ford Motor Company for supporting this project.

REFERENCES

[1] R.Hirsh, G.N. DeSouza, and A.C. Kak, "An iterative approach to the hand-eye and base-world calibration problem," in Proceedings of 2001 IEEE International Conference on Robotics and Automation, Vol. 1, pp.2171-2176, May 2001.

[2] G.N. DeSouza, A Subsumtive, Hierarchical, and Distributed Vision-based Architecture for Smart Robotics, Ph.D. Dissertation., School of Electrical and Computer Engineering, Purdue University, May 2002.

[3] A.Kosaka and A.C. Kak, "Stereo vision for Industrial application," Handbook of Industrial Robotics, edit. S.Y. Nof, pp. 269-294, John Wiley & Son, Inc., NY, 1999.

[4] S.K. Nayar, H. Murase, and S.A. Nene, "Learning, Positioning, and Tracking Visual Appearance," in 1994 IEEE International Conference on Robotics and Automation, Vol. 4, May 1994, pp.3237-3244.

[5] J.L. Edwards, "An Active, Appearance-based approach to the Pose Estimation of Complex Objects," in 1996 IEEE/RSJ International Conference on Intelligent Robots and System , Vol. 3, Nov. 1996, pp. 1458-1465.

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

[7] 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.

[8] T.F. Cootes, G.J. Edwards, C.J. Taylor, "Active appearance models," in Proc. European Conference on Computer Vision 1998 (H.Burkhardt & B. Neumann Ed.s). Vol. 2, pp. 484-498, Springer, 1998.

[9] T.F. Cootes, C.J. Taylor, Statistical models of appearance for computer vision, in WWW publication, October 2001, Available from http://www.isbe.man.ac.uk/bim/refs.html

[10] G.D. Hager, and P.N. Delhumeur, "Efficient Region Tracking with Parameteric Models of Geometry and Illumination," IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 20, pp. 1025-1039, Oct. 1998.

[11] M. Gleicher, "Projective Registration with Difference Decomposition," in IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Jun 1997, pp.331-337

[12] M.L. Cascia, S.Sclaroff, and V. Athitsos, "Fast, Reliable Head Tracking under Varying Illumination: An Approach Based on Registration of Texture-Mapped 3D Models," IEEE Trans. Pattern Analysis and Machine Intelligence, Vol.22, pp. 322-336, Apr 2000.

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

[14] Y.Yoon, G.N. DeSouza, and A.C. Kak, "Real-time Tracking and Pose Estimation for Industrial Objects using Geometric Features," in 2003 IEEE International Conference on Robotics and Automation, Sep 2003, pp.3473-3478.

[15] O.D. Faugeras, Three-Dimensional Computer Vision, MIT Press, 1993.

[16] M. B. Stegmann, B.K. Ersboll, R. Larsen, "FAME--A Flexible Appearance Modeling Environment," IEEE Trans. Medical Imaging, Vol. 22, No. 10, pp. 1319-1331, Oct 2003.

[17] M. B. Stegmann, "Object tracking using active appearance models," Proc. 10th Danish Conference on Pattern Recognition and Image Analysis, vol. 1, pp. 54-60, DIKU, 2001.

[18] 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.