An efficient iterative pose estimation algorithm

S.H. Or[1] W.S. Luk[2] K.H. Wong[1] I. King[1]

shor@cse.cuhk.edu.hk Wai-Shing.Luk@cs.kuleuven.ac.be khwong@cse.cuhk.edu.hk king@cse.cuhk.edu.hk

[1]Department of Computer Science & Engineering
The Chinese University of Hong Kong

[2]Departement Computerwetenschappen, Katholieke Universiteit Leuven

Abstract. We propose a novel model-based algorithm which finds the 3D pose of an object from an image by breaking down the estimation process into two linear processing stages, namely the depth recovery and the pose calculation. The depth recovery stage determines the new positions of the model point set in 3D space whereas the pose calculation step is a least-square estimation of the transformation parameters between the point set formed from the previous stage and the model set. The estimates are iteratively refined until converged. The advantage of using our algorithm is that the computational cost is much reduced. We test our algorithm by applying it to both synthetic as well as real time head tracking problem with satisfactory results.
index : Pose Estimation, Real time vision, Human-computer Interface.

1 Introduction

Determining the pose (position and orientation) of a moving object from an image sequence is useful in applications such as photogrammetry, passive navigation, industry inspection and human-computer interfaces, etc. In this paper, we are interested in the model-based pose estimation algorithm. A number of work have been done by previous researchers [13, 7, 4, 3]. Some of them are optical flow based[4], which require massive computation power and are not suitable for real time applications. Others use nonlinear iterative methods such as Newton's method, which is sensitive to the initial guess supplied. Our interest is in developing an efficient algorithm which is suitable for real time applications on human motion anaylsis. This area is characterized by the rapid motion of the object concerned and high probability of occlusion/re-appearance of part of the object in the subsequent frames.

2 Problem formulation

We take the assumption of the pin-hole camera model with full perspective projection as shown in Fig. 1. f is the focal length of the camera and the focal plane of the camera is placed at distance f from the camera origin. The problem of pose estimation can be described as follows. Given a 3D point set $\{\mathbf{P}_i\}; i \in 1, 2, \ldots, N$ where $\mathbf{P}_i = (X_{P_i}, Y_{P_i}, Z_{P_i})^t, i \in 1, 2, \ldots, N$ are the coordinates of the

model points at a reference instant. Assuming a rigid transformation is being applied to this point set, yielding a new set which then projected onto an image plane, producing $\{Q_i\} = (X_{Q_i}, Y_{Q_i}) \in 1, 2, \ldots, N$, we seek R, $\mathbf{T}$ and $\{d_i\}; i \in 1, 2, \ldots, N$ such that

$$e^2(R, \mathbf{T}, \{d_i\}) = \min\left(\sum \|d_i \hat{\nu_i} - (R\mathbf{P}_i + \mathbf{T})\|^2\right) \quad (1)$$

e^2 is a measurement function of the current fit of rotation R, translation $\mathbf{T}$, and $\{d_i\}$. d_i corresponds to the depth which determines where in 3D space should the actual point be located along the projection ray. $\hat{\nu_i} = (X_{Q_i'}^2 + Y_{Q_i'}^2 + f^2)^{-\frac{1}{2}}(X_{Q_i'}, Y_{Q_i'}, f)$ is the unit vector on this ray.

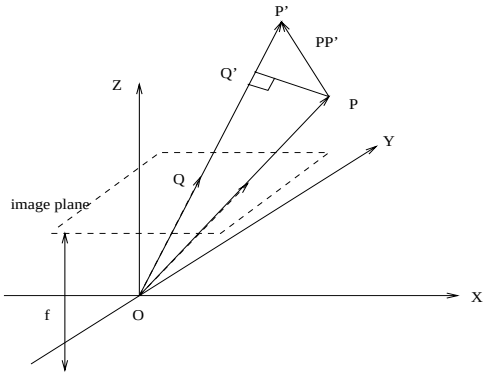

Fig.1 Relationship between original point and projected point

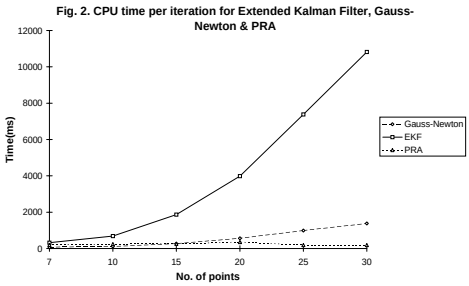

Fig.2 CPU time per iteration for Extended Kalman Filter, Gauss-Newton and PRA

In the formulation above, least square minimization methods can be applied to the $N+6$ parameter space to find the minimum of the measurement function e^2 where N is the number of feature points of the object. This approach is adopted by [9, 5]. Since the original 3D coordinates are being transformed by a perspective projection, the objective function is thus nonlinear. To solve for the solution, iterative methods are required which is usually a time consuming process. In addition, the computational effort needed increases rapidly as the number of points increases.

3 Our Algorithm

Recently there are some work which focus on using the projection rays as a guide and perform the pose estimation in 3D space [8]. However, the approach used the iterative point matching method, which demands for much computational effort. This will not suit the requirement of a real time computer-human interface, which is what we are interested in. Inspired by the inverse projection ray approach, we propose that the minimization process be broken down into two stages: The first

stage will estimate the position of all the feature points in the 3D space, i.e. $\{d_i\}$. The estimated point set will be passed to the second stage, which is a least square fitting of the model set and estimated point set. The above procedure is repeated until the result converges. By dividing the estimation process into two stages, the size of the solution space in each stage is much reduced and the cost in locating the solution decreased significantly. The resulting algorithm is very efficient. Our algorithm basically works in the following way:

1. Assume the object has no motion, determine $\{d_i\}$, $i \in 1, 2 \ldots, N$ and $\mathbf{T}$;
2. Assume $\{d_i\}$, $i \in 1, 2, \ldots, N$ are determined, estimate R and $\mathbf{T}$.
3. Update the state of the solution and iterate until the values of R and $\mathbf{T}$ less than some thresholds.

Note that we estimate both $\{d_i\}$ and $\mathbf{T}$ in the first stage. The point of recovering $\mathbf{T}$ earlier is to make our algorithm more robust since leaving $\mathbf{T}$ to be determined in later stage will easily lead to a result of stucking in the local minimum as indicated in some of our experiments. In the second stage, we will minimize the following objective function

$$e^2(R, \mathbf{T}) = \sum \|\mathbf{Q}'_i - (R\mathbf{P}_i + \mathbf{T})\|^2 \tag{2}$$

where $\mathbf{Q}'_i$ is the estimated transformed point set in the previous estimation stage, $\mathbf{P}_i$ is the original(model) point. Various efficient algorithms [2, 6, 11] are available for the fitting of two point sets. We choose the singular value decomposition method[6, 11] due to its robustness in noise handling and that only a 3×3 matrix decomposition procedure is needed.

3.1 Complete algorithm

The complete algorithm is as follows:

1. Minimize the function below to estimate $\{d_i\}$, $i \in 1, 2, \ldots, N$:

   $$\sum \|d_i\hat{\nu}_i - (\mathbf{P}_i + \mathbf{T})\|^2$$

   The resulting $\{d_i\}$, $i \in 1, 2, \ldots, N$ and $\mathbf{T}$ is given by:

   $$d_i = \hat{\nu_i}^t(\mathbf{P}_i + \mathbf{T}) \tag{3}$$

   $$\mathbf{T} = -\left(\sum_{i=1}^{N} A_i\right)^{-1}\left(\sum_{i=1}^{N} A_i\mathbf{P}_i\right) \tag{4}$$

   where A_i is a 3 by 3 matrix given by

   $$A_i = I - \hat{\nu}_i\hat{\nu}_i^t$$

   Please refer to appendix for the derivation of the above formula.
2. Using the estimated $\{d_i\}$ $i \in 1, 2, \ldots, N$ in the previous stage, apply the SVD method to determine the rigid transformation R and $\mathbf{T}$.

3. Update $\mathbf{P}_i$ $i \in 1, 2, \ldots, N$ by $\mathbf{P}_i \leftarrow R\mathbf{P}_i + \mathbf{T}$.

Note that A_i's are not needed to be stored explicitly. Firstly, each A_i depends only on $\hat{\nu}_i$, which is fixed before the execution of our algorithm. Therefore the value of $(\sum A_i)^{-1}$ in Eq. (4) can be pre-computed before the iterations, which avoids the expensive inverting operation of a matrix. Moreover, the matrix vector product $A_i P_i$ can be computed efficiently using the formula $\mathbf{P}_i - \hat{\nu}_i(\hat{\nu}_i^t \mathbf{P}_i)$. We also noticed the algorithm by DeMenthon and Davies [3] which requires relatively economic arithmetic operations. However, our argument for the efficiency is still valid by the following reasons. Firstly, in the approach used by DeMenthon and Davies, the algorithm requires an approximate pose estimate from the previous stage, which would inevitably contribute to the computational requirement. Moreover, their method does not guarantee the orthonormality of the resulting rotation matrix. Whereas our algorithm used the singular value decomposition, which provides an orthonormal result, hence improves the accuracy and thus justifies the increase in computation.

4 Performance comparison

We have compared the performance of two established approaches with our algorithm. The main interest here is the computational efficiency since the main concern during the motion tracking application would probably be the timing requirement as well as accuracy and stability. We chose Gauss-Newton method(By Lowe[7]) and Extended Kalman Filter(By Broida *et al.* [10] and Azarbayejani *et al.*[1]) method due to their robustness and computational efficiency. Lowe's approach, the Extended Kalman filter as well as our algorithm were implemented in C.

All the simulations are performed on a SUN Ultra 1/170 workstation. The same data set was used which involved both translation and rotation of two to four planes in 3D space (10 points per plane are used). The motion trajectory consists of increasing values in both rotation and translation for 100 frames. Monte Carlo simulation was applied to the data and Gaussian random noise with zero mean and standard deviation of one unit is added to the coordinates of each feature point. The plots of average time per frame used by various approaches are shown in Fig. 2. In the plot, our algorithm is represented using the title "PRA" which stands for "*Projection Ray Attraction*". The time unit in our experiments is milliseconds. The computational advantage of our approach is clear from the figure. It can be seen that the time required by Lowe's approach increases linearly with the number of points whereas our approach remains roughly the same. The main reason is that Gauss-Newton method requires the solution of a $(N+6) \times (N+6)$ matrix, where N is the number of feature points measured. This matrix inverse operation is surely computationally the most demanding step. The long computation time of that of Extended Kalman filter stems from the fact that it has to solve for a $2N \times 2N$ matrix.

5 Synthetic Data Experiment

The performance of our algorithm under noisy environment is investigated. Two sets of experiments are being performed. We first describe the setup since they are common for both experiments. A number of points are randomly generated in the 3D space such that they will project on an image plane of size 2 by 2. These model points are then transformed as follows: a rotation about the axis $(1, 1, 1)$ by $6°$ followed by a translation of $(5.0, 3.0, 6.0)$. The transformed points are then projected on the image plane. The image coordinates are then digitized to a screen resolution of 512×512. The process of digitization introduces noise in this case. 100 random scenes are generated according to this setup and the digitized coordinates are used together with the original generated 3D model to determine the pose.

The errors shown in the following are all relative errors. We used quaternion to represent the rotation such that all the estimated results are in vector form. The relative error of a vector is defined by the Euclidean norm of the difference between the estimated and the true vector divided by the Euclidean norm of the true vector. The first set of experiments tests the performance of the algorithm under different number of points in the scene. The results are shown in Fig. 3a. From the plot, it is observed that the estimated rotation is relatively stable with respect to the variation of the number of points whereas the translation has the greatest improvement from 8 to 12 points. In addition, the digitization only introduces a small effect on the estimated results, as can be seen in the range of errors it is only within 3 percents.

Another set of experiments simulates the measurement noise by introducing an offset value to both digitized x and y coordinates. These offset values are normal distributed with variance of k pixels where k is the parameter to be varied in the experiments. 16 points are used in this set of experiments. Fig. 3b shows the results and our algorithm again is quite stable under the situation of increasing noise.

6 Real image testing

Pose tracking involves continuous monitoring of the pose of an object from the input image sequence. Our algorithm can be easily adapted for continuous tracking by performing estimation between successive image frames. We test our algorithm by applying it to track the head of a person in an image sequence. Our implementation is done on an SGI Indy workstation with MIPS R4400 processor. To verify the correctness of the recovered pose informaton, we reproduce the same motion on the workstation at the same time. Though it is difficult in this case to estimate the performance figure of the algorithm, an overall estimate of the usefulness of the algorithm can be obtained. We used the structure from motion approach to obtain the data set for the face to be tracked, following the algorithm by Szeliski [9]. In our experiment, only twelve points are used. All points are selected on the criteria that they can be tracked unambiguously

throughout the sequence so as to increase the accuracy of the recovered pose. We use four points for the head boundary including ears, four points for the mouth, and four points for each of the eyebrows.

Before tracking, the user has to initialize the tracker. The selected feature points should be the same as those taken in the calibration stage. These selected feature points are then tracked by the normalized correlation method to give coordinates in successive frames. It is inevitable that error in selecting feature exists due to disagreement in the location of the feature selected and that in the calibration stage. However, according to our experience, this discrepancy is minor in that our algorithm can still maintain the tracking over a long period, say over 100 frames. The tracking information(rotation and translation) will be used to control a synthetic head model[1]. Results are shown in Fig. 4.

7 Discussion

In the experiments on head tracking, we found that our algorithm can effectively recover the motion information from the images. The main advantage of our algorithm is that it is computationally efficient. In addition, by performing the fitting of the model set in 3D object space and not on the image plane, a larger range of convergence is achieved which enables our algorithm to be more stable with respect to different initial guesses. However, our algorithm also has the following drawbacks. Firstly, due to the fact that our algorithm is formulated under the situation of two frames only, thus the error in the estimation will accumulate in the subsequent frames and may lead to failure after a large number of frames. Moreover, in the current formulation, the problem of occlusion is not being handled.

For the first problem, one solution is to bootstrap the algorithm itself after a number of frames. This is possible since our algorithm has quite a large range of convergence (approximately 40 degrees for the rotation angles from our experiments). Another solution is to cross-check the pose estimaton result with two or more frames earlier so as to minimize the error itself. The second problem demands a further enhancement of our algorithm to handle the occlusion problem, which is not that difficult due to the model based nature of our algorithm. We are currently working in this direction because the task of human motion tracking including head tracking is a primary goal of our project. An exciting application of head tracking is the usage of very low bit rate transmission in video conferencing.

8 Conclusion

An efficient pose estimation algorithm is presented. By breaking down the pose estimation process into two separate linear stages, the computation requirement

[1] The synthetic character is a public domain implementation of the facial animation work by Keith Waters[12]

is significantly reduced. Tests on synthetic data as well as real world head tracking have demonstrated the effectiveness of our algorithm.

Appendix: Derivation of Initial Guess

We want to minimize the function below to establish the initial guess:

$$\Theta = \sum_{i=1}^{N} \|d_i \hat{\nu_i} - (\mathbf{P}_i + \mathbf{T})\|^2$$

Applying partial differentiation to Θ with respect to d_i and $\mathbf{T}$ respectively and setting them equal to zero, we have

$$\frac{\partial \Theta}{\partial d_i} = 0 \Rightarrow d_i = \hat{\nu_i}^t(\mathbf{P}_i + \mathbf{T})$$
$$\frac{\partial \Theta}{\partial \mathbf{T}} = 0 \Rightarrow \mathbf{T} = \frac{1}{N}\sum_i (d_i \hat{\nu_i} - \mathbf{P}_i)$$

$$\mathbf{T} = \frac{1}{N}\sum([\hat{\nu_i}\hat{\nu_i}^t]\mathbf{P}_i - \mathbf{P}_i) + \frac{1}{N}\sum_i([\hat{\nu_i}\hat{\nu_i}^t]\mathbf{T})$$

$$\Rightarrow (N\mathbf{I} - \sum \hat{\nu_i}\hat{\nu_i}^t)\mathbf{T} = \sum_i (\hat{\nu_i}\hat{\nu_i}^t - \mathbf{I})\mathbf{P_i}$$

Writing A_i as $\mathbf{I} - \hat{\nu_i}\hat{\nu_i}^t$, we have

$$\mathbf{T} = -(\sum_{i=1}^{N} A_i)^{-1}(\sum_{i=1}^{N} A_i \mathbf{P}_i)$$

References

1. A.Azarbayejani and A. Pentland. Recursive estimation of motion, structure, and focal length. *IEEE Trans. Pattern Anal. Machine Intell.*, 17(6):562–575, June 1995.
2. B.K.P.Horn. Closed-form solution of absolute orientation using unit quaternions. *J. Opt. Soc. Amer.*, 4:629–642, April 1987.
3. D.F.Dementhon and L. S. Davis. Model-based object pose in 25 lines of code. *Intl. Journal of Comput. Vision*, 15:123–141, 1995.
4. H.Li, P.Roivainen, and R.Forchheimer. 3-d motion estimation in model-based facial image coding. *IEEE Trans. Pattern Anal. Machine Intell.*, 15(6):545–555, June 1993.
5. J.Weng, N.Ahuja, and T.S.Huang. Optimal motion and structure estimation. *IEEE Trans. Pattern Anal. Machine Intell.*, 15(9):864–884, September 1993.
6. K.S.Arun, T.S.Huang, and S.D.Blostein. Least-square fitting of two 3-d point sets. *IEEE Trans. Pattern Anal. Machine Intell.*, 9(5):698–700, September 1987.

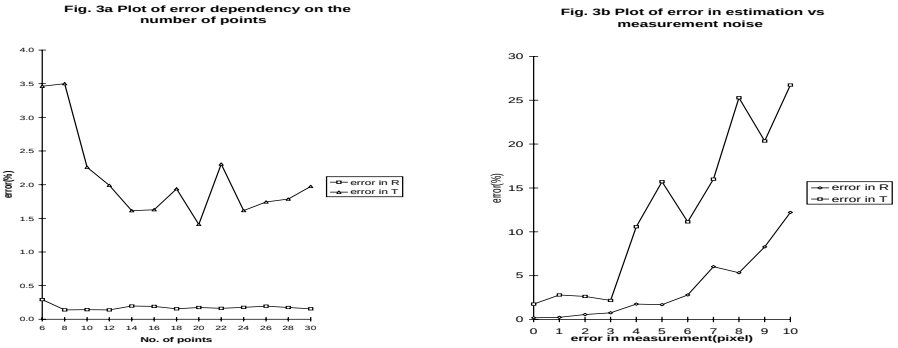

Fig. 3. Left) Plot of error dependencey on the number of points, right) Plot of error in estimation vs. measurement noise

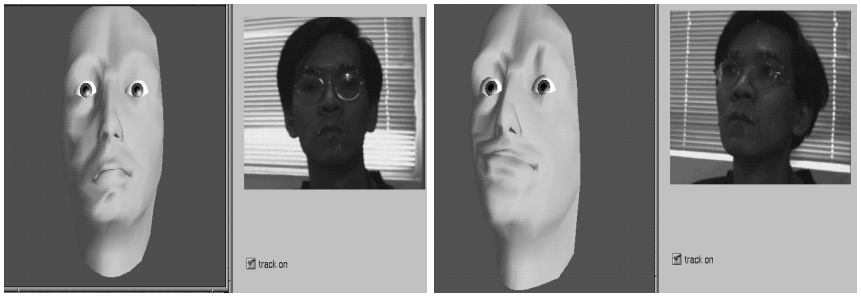

Fig. 4. Sample run of the head tracking application. Left) initialisation, Right) Rotation of the head and reconstructed pose.

7. D. G. Lowe. Fitting parameterized three dimensional models to images. *IEEE Trans. Pattern Anal. Machine Intell.*, 13(5):441–450, May 1991.
8. P.Wunsch and G.Hirzinger. Registration of cad-models to images by iterative inverse perspective matching. *Proc. ICPR 96*, pages 78–83, November 1996.
9. R.Szeliski and Sing Bing Kang. Recovering 3d shape and motion from image streams using non-linear least squares. *Cambridge Research Laboratory Technical Report*, March 1993.
10. T.J.Broida. Recursive 3-d motion estimation from a monocular image sequence. *IEEE Trans. Aerospace Electronic Systems*, 26(4):639–655, July 1990.
11. Shinji Umeyama. Least-square estimation of transformation parameters between two point pattern. *IEEE Trans. Pattern Anal. Machine Intell.*, 13(4):376–380, April 1991.
12. K. Waters. A muscle model for animating three-dimensional facial expression. *Comput. Graphics*, 21(4):17–24, 1987.
13. Z. Zhang and O. Faugeras. *3D Dynamic Scene Analysis*. Springer-Verlag, 1992.

This article was processed using the LaTeX macro package with LLNCS style