A Novel Point-based Pose Estimation Algorithm

Siu-hang Or
shor@cse.cuhk.edu.hk

Kin-hong Wong
khwong@cse.cuhk.edu.hk

Irwin King
king@cse.cuhk.edu.hk

The Chinese University of Hong Kong
Computer Science and Engineering Department
Shatin, N.T., Hong Kong

Abstract

Pose estimation and motion tracking from image sequences is useful in many robotic applications. We propose a real time algorithm which performs these two tasks using an iterative two-stage process: (1) depth prediction and (2) pose calculation. An analytical expression for determining the convergence is also obtained. The algorithm was tested successfully on synthetic and real image sequences.

1. Introduction

Determining the motion information from an image sequence is useful in various applications such as photogrammetry, passive navigation, industry inspection, and computer-human interface. The Kalman filter method [8, 3] provides an optimal least square estimate to the motion information. However, it has difficulties in handling the occlusion problem nicely. Moreover, the Kalman filter method has the numerical instability problem which requires special treatment [5]. In this paper, we propose a new algorithm for pose/motion estimation which provides a least square estimate to the motion recovery problem and is not based on the Kalman filter methodology. The framework is general enough to be able to incorporate more robust techniques to increase its stability. Also an extension, which taking the advantage of the model-based nature of our algorithm, to solve the occlusion problem is currently under development.

Consider a camera with its focal plane's origin placing at $(0, 0, f)$, as in Fig. 1, where f is the focal length of the camera. Given a set of 3D points $\{\mathbf{P}_i\}$ where $i = 1, 2, \ldots, N$ which is being transformed (rotation followed by translation) to yield another set of points $\{\mathbf{P}'_i\}$; $i = 1, 2, \ldots, N$ ($\mathbf{P}'_i$ is unknown in our problem). The projection of the transformed point set on the image plane is $\{Q'_i : (x_{Q'_i}, y_{Q'_i}), i = 1, 2, \ldots, N\}$.[1] Assuming the coordinates of the set of transformed points are $\{\mathbf{P}'_i : (x_{P'_i}, y_{P'_i}, z_{P'_i})^t\}$, they are related to the image points by a perspective projection $x_{Q'_i} = f * \frac{x_{P'_i}}{z_{P'_i}}$, $y_{Q'_i} = f * \frac{y_{P'_i}}{z_{P'_i}}$. In this arrangement, any point in the image would give an inverse projection ray with $\hat{\nu}_i = (x^2_{Q'_i} + y^2_{Q'_i} + f^2)^{-\frac{1}{2}}(x_{Q'_i}, x_{Q'_i}, f)^t$ as the direction vector on this ray. The actual coordinates of this point in the 3D space can be written as $\mathbf{P}'_i = d_i\hat{\nu}_i$, where d_i(a scalar) is the depth of the actual feature point from the perspectivity center.

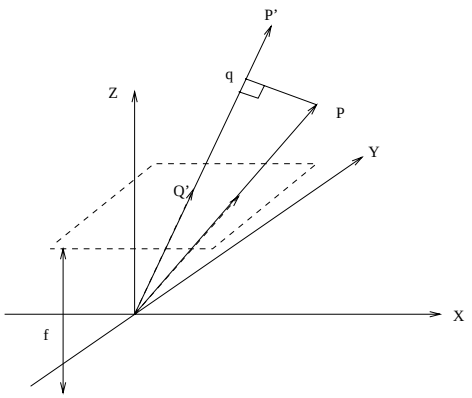


Figure 1. Relationship between original, transformed and projected points

Given $\{Q'_i\}$ and a three dimensional model $\mathbf{P}_i = (x_{P_i}, y_{P_i}, z_{P_i})^t$; $i = 1, 2, \ldots, N$, we seek R, $\mathbf{T}$ and

[1] We used bolded font for a 3D vector in the camera coordinate space and normal font for a 2D vector on the image plane.

$\{d_i\}; i = 1, 2, \ldots, N$ such that

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

where e is a measurement function of the current fit of the rotation matrix R and translation $\mathbf{T}$. And d_i corresponds to the depth value which determines where in the 3D space should the actual point be located along the projection ray.

2 Pose estimation algorithm

To solve the above minimization problem, least square minimization methods can be applied to the whole parameter space to find the minimum of the measurement function e. However as the number of parameters increases, the computational effort required to find the minimum increases rapidly due to the non-linear nature of the problem. With the observation that the actual feature points must reside on the projection rays, we propose that the minimization process should be broken down into a two-stage process: the first stage estimates the positions of all feature points in the 3D space. The estimated positions will then be passed to the second stage - a least square fitting of two 3D point sets, which can be efficiently solved by various established non-iterative methods such as the singular value decomposition method [1, 4]. The above procedure is repeated until the values of R, and $\mathbf{T}$ converge. The algorithm is as follows:

Algorithm 1 Model-based Pose Estimation

▷ *Input*

$\{\mathbf{P}_i\}; i = 1, 2, \ldots, N$: 3D description of the original object(model),

$\{Q_i' : (x_{Q_i'}, y_{Q_i'})\}; i = 1, 2, \ldots, N$: transformed feature point coordinates on the image plane.

▷ *Procedure*

1 **while** (change in R or $\mathbf{T}$ not less than some threshold values)

2 Estimate $\{d_i\}; i = 1, 2, \ldots, N$ in Eq.(1). $\{d_i\}$ is estimated by $|\mathbf{P}_i \cdot \hat{\nu}_i|$ **where** $\hat{\nu}_i$ is the unit vector along the projection ray formed by image point Q_i' and the origin.

3 Perform a least square fitting to Eq. (1) to estimate R, and $\mathbf{T}$ by the singular value decomposition method [4].

4 Update $\mathbf{P}_i$ by $\mathbf{P}_i \leftarrow R\mathbf{P}_i + \mathbf{T}$.

end while

As seen in Fig. 1, the estimated position of the feature point in the 3D space is only the perpendicular projection of the model point on the projection ray. After each iteration, we will get a better estimation and the model will be transformed closer to the projection rays by the updating step in our algorithm. According to Eq. (1), this corresponds to a reduction in the objective function value at each iteration. Eventually our algorithm will converge to the solution in which the sum of the squares of the perpendicular distances between the model and the actual locations is a minimum or zero.

3 Convergence analysis

The analysis of the convergence condition of a particular algorithm is usually neglected in most pose estimation literatures. In this paper, we will try to estimate the effect of different motion parameters on the recovered values. Assume a model point $\mathbf{P}$ is being transformed to another location $\mathbf{P}'$ by a rigid body transform, the perpendicular projection of $\mathbf{P}$ onto $\hat{\nu}$ is $\mathbf{q}$ (see Fig. 1.), therefore

$$\mathbf{q} = \mathbf{P}'(1 - \frac{\mathbf{P}\mathbf{P}' \cdot \mathbf{P}'}{\|\mathbf{P}'\|^2}).$$

The second term on the right hand side of the equation, which is the difference between the estimated position and the true one, can be treated as an error term $\varepsilon = \frac{\mathbf{P}\mathbf{P}' \cdot \mathbf{P}'}{\|\mathbf{P}'\|^2}$. This error term can affect the recovered transformation parameters in the second stage of our algorithm. For a large value of ε (say $\varepsilon = 0.5$, which corresponds to shifting the point along the projection ray by a significant amount), the estimated transformation would align the object farther away from the original position such that our algorithm will not converge to the correct result in the subsequent iterations. Substituting $\mathbf{P}'$ by $R\mathbf{P} + \mathbf{T}$ in the error term, we have

$$\varepsilon = \frac{R\mathbf{P} \cdot R\mathbf{P} + R\mathbf{P} \cdot \mathbf{T} + \mathbf{T} \cdot (\mathbf{T} - \mathbf{P}) - \mathbf{P} \cdot R\mathbf{P}}{R\mathbf{P} \cdot R\mathbf{P} + 2R\mathbf{P} \cdot \mathbf{T} + \mathbf{T} \cdot \mathbf{T}}. \quad (2)$$

For the correct convergence of our algorithm, the above equation suggests that the difference between $\mathbf{T}$ and $\mathbf{P}$ should be small. In order to find the relationship between ε and input Euler angles, we plot the resulting ε against the variation of one of the angles while others are kept fixed. The resulting plots are shown in Fig. 2. It is found that the error increases rapidly after one of the input angles exceeds 20 degrees.

4 Synthetic and real data experiment

Experiment using synthetic data - To test the validity of our approach, a randomly generated point set of 8 points inside a unit cube is used. First the point set is transformed(rotate and translate) and projected on an image plane. Then our algorithm is applied to recover the transformation parameters. We vary one of the parameters, say pitch angle, while keeping the other parameters at zero. The resultant plots of recovered Euler angles against corresponding input ones are shown in Fig. 3. This experiment confirms the result estimated by Eq. (2) and Fig. 2. That is our algorithm becomes unreliable after one or more of the input angles exceed 20 degrees. Hence we conclude that the convergence requirement under different values of R, $\mathbf{T}$ and $\mathbf{P}$ can be estimated by Eq. (2).

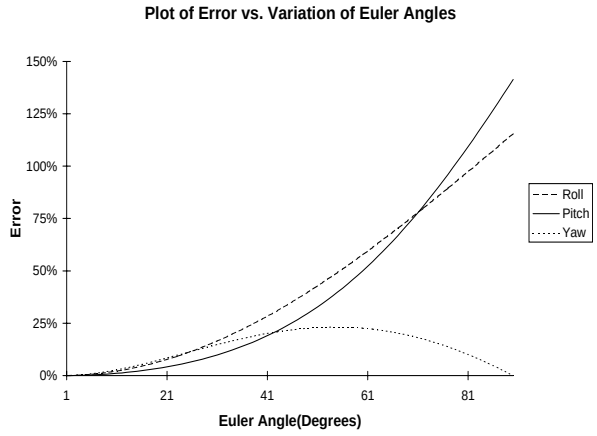


Figure 2. Plot of error vs. variation of Euler Angles

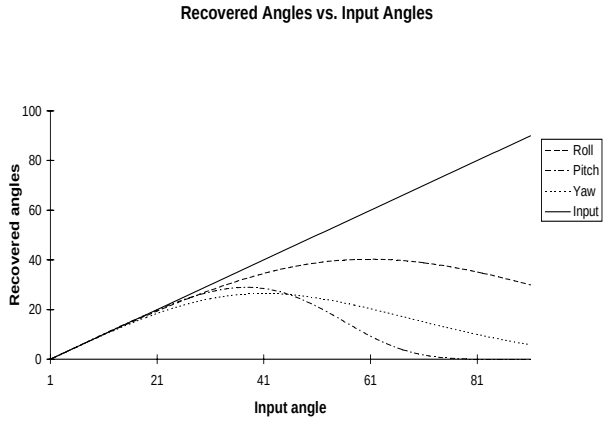


Figure 3. Plot of estimated Euler angles vs. input angles

Experiment using real data - We also use our algorithm to track the movement of a person's head in front of a camera. The implementation is done on an SGI Indy workstation. Some preliminary results are shown in Fig. 4. During initialisation, as shown in Fig. 4a, the user first selects the invariant features, such as mouth corners, eyebrows, etc. as the points to be tracked. The animated face to be controlled is shown on the left of the display at the same time. The feature points are then tracked automatically by the normalized correlation method to give coordinates in successive frames. Our algorithm can thus estimate the motion of the human head and the results are used to control a computer generated human face[2].[2] A shot made at the middle of the run is shown in Fig. 4b. It can be seen that the head of the user made some rotation about the X-axis and Y-axis to give a pose of looking up, and the pose can be reproduced by the animated face accordingly. The pose information generated by the whole sequence is recorded and plotted in Fig. 4c and Fig. 4d. In Fig. 4c, the plots of Roll(rotation about Z-axis), Pitch(rotation about X-axis) and Yaw(rotation about Y-axis) are shown for the whole run. In Fig. 4d, the estimations for the translation are plotted. Since the computation cost is rather low, our algorithm is capable of running in real time in the above test.

5 Conclusion

A real time pose estimation algorithm is developed using a two-stage iterative method. A convergence analysis is performed and the algorithm has been tested by synthetic as well as real data with satisfactory results.

References

[1] K.S.Arun and 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, Sept. 1987.

[2] K. Waters. A muscle model for animating three-dimensional facial expression. *Comput. Graphics*, 21(4):17–24, 1987.

[3] T.J.Broida. Recursive 3-d motion estimation from a monocular image sequence. *IEEE Trans. Aerospace Electronic Systems*, 26(4):639–655, July 1990.

[4] Shinji Umeyama. Least-square estimation of transformation parameters between two point pattern. *IEEE Trans. Pattern Anal. Machine Intell.*, 13(4):376–380, Apr. 1991.

[5] Z. Zhang and O. Faugeras. *3D Dynamic Scene Analysis*. Springer-Verlag, 1992.

[6] 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] The codes for the computer synthesized human face is a public domain implementation by Keith Waters.

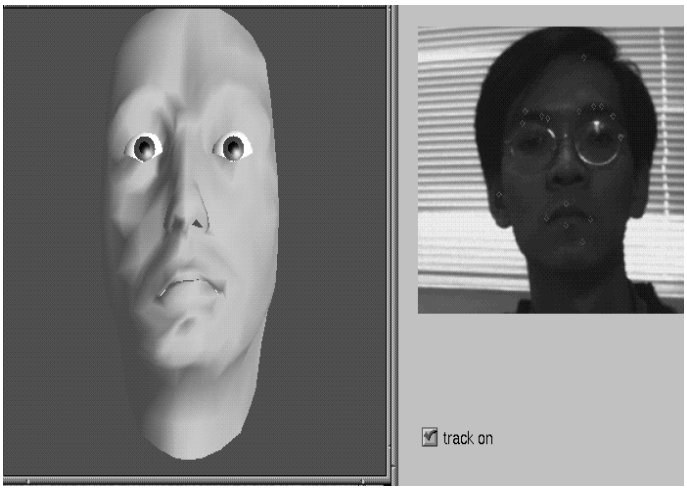
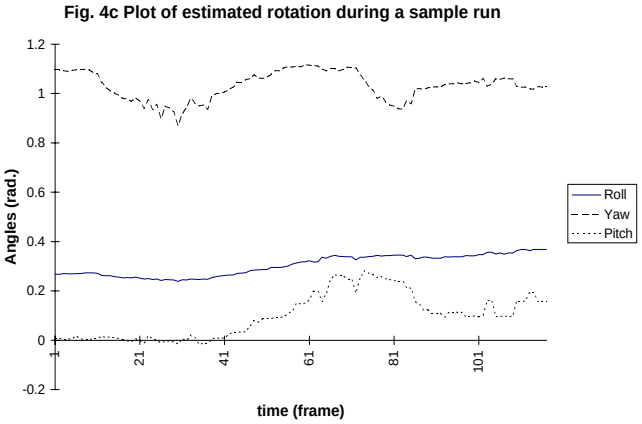


Figure 4. Sample run of the head tracking application. a) initialisation,



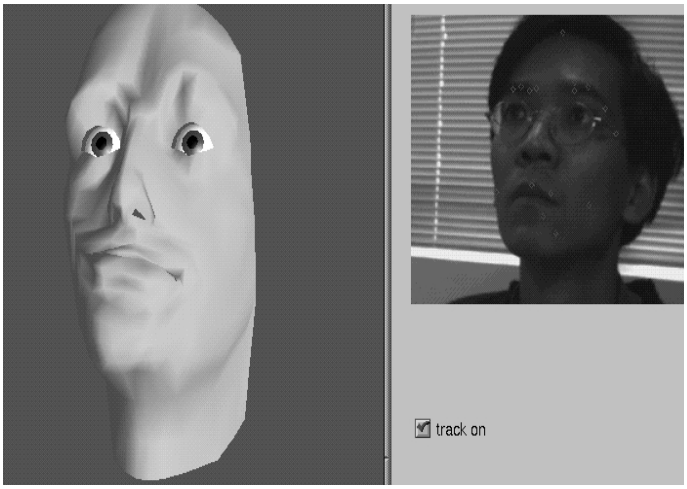


Figure 4. b) rotation of the head,