

Robust Head Motion Computation by Taking Advantage of Physical Properties

Zicheng Liu, Zhengyou Zhang

Microsoft Research, One Microsoft Way, Redmond, WA 98052, USA

Abstract

We present a new algorithm to compute the head motion between two views from the correspondences of five feature points (eye corners, mouth corners, and nose top), and zero or more other image point matches. The algorithm takes advantage of the physical properties of the feature points such as symmetry to significantly improve the robustness of head motion estimation. This idea can be easily extended to any number of feature point correspondences.

1 Introduction

In this paper, we present a new algorithm to compute the head motion between two views from the correspondences of five feature points including eye corners, mouth corners and nose top, and zero or more other image point matches. This is useful for some face modeling and tracking systems where the feature points can be obtained. For example, the user can mark the feature points on the two images [5], or the user marks the feature points on one image while their correspondences are tracked on the other image [2], or the feature points are extracted and tracked automatically [3].

If the image locations of these feature points are precise, one could use five-point algorithm to compute camera motion. However, this is usually not the case in practice. A human in general cannot mark the feature points with high precision. A tracking algorithm may not result in perfect matches either. When there are errors, a five-point algorithm is not robust even when refined with a bundle adjustment technique. The key idea of our work is to use the physical properties of the feature points to improve robustness. We use the property of symmetry to reduce the number of unknowns. We put reasonable lower and upper bounds on the nose height and represent the bounds as inequality constrains. As a result, the algorithm becomes significantly more robust.

Even though in this paper we only describe our algorithm in the case of five feature points, it is straight-

forward to extend the idea to any number (less than or bigger than 5) of feature points for robustness improvement.

In Section 2, we describe the motion estimation algorithm from five feature points only. In Section 3, we extend the algorithm to incorporate other image point matches obtained from image registration methods. The experiment results are shown in Section 4.

2 Head motion estimation from five feature points

We use E_1 , E_2 , M_1 , M_2 , and N to denote the left eye corner, right eye corner, left mouth corner, right mouth corner, and nose top, respectively (See Figure 1). Denote E as the midpoint of E_1E_2 and M the midpoint of M_1M_2 . Notice that human faces exhibit some strong structural properties. For example, left and right sides are very close to being symmetric about the nose; eye corners and mouth corners are almost coplanar. We therefore make the following reasonable assumptions: NM is perpendicular to M_1M_2 , NE is perpendicular to E_1E_2 , and E_1E_2 is parallel to M_1M_2 .

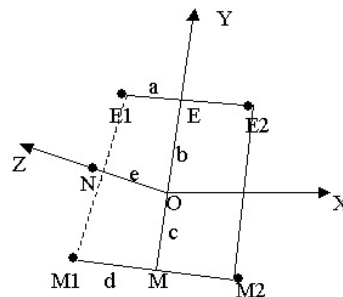


Figure 1: The new coordinate system Ω_0 .

Let π be the plane defined by E_1 , E_2 , M_1 and M_2 . Let O denote the projection of point N on plan π . Let Ω_0 denote the coordinate system which is originated at O with ON as the Z -axis, OE as the y -axis. In this coordinate system, based on the assumptions mentioned earlier, we can define

the coordinates of E_1, E_2, M_1, M_2, N as $(-a, b, 0)^T, (a, b, 0)^T, (-d, -c, 0)^T, (d, -c, 0)^T, (0, 0, e)^T$, respectively. Thus, we only need 5 parameters to define these five points in this local coordinate system, instead of 9 parameters for generic five points.

Let \mathbf{t} denote the coordinates of O under the camera coordinate system, and \mathbf{R} the rotation matrix whose three columns are vectors of the three coordinate axis of Ω_0 . For each point $\mathbf{p} \in \{E_1, E_2, M_1, M_2, N\}$, its coordinate under the camera coordinate system is $\mathbf{R}\mathbf{p} + \mathbf{t}$. We call (\mathbf{R}, \mathbf{t}) the *head pose transform*.

Given two images of the head under two different poses (assume the camera is static), let (\mathbf{R}, \mathbf{t}) and $(\mathbf{R}', \mathbf{t}')$ be their head pose transforms. For each point $\mathbf{p}_i \in \{E_1, E_2, M_1, M_2, N\}$, if we denote its image point in the first view by \mathbf{m}_i and that in the second view by \mathbf{m}'_i , we have the following equations:

$$proj(\mathbf{R}\mathbf{p}_i + \mathbf{t}) = \mathbf{m}_i \quad (1)$$

and

$$proj(\mathbf{R}'\mathbf{p}_i + \mathbf{t}') = \mathbf{m}'_i \quad (2)$$

where *proj* is the perspective projection. Notice that we can fix one of the a, b, c, d, e since the scale of the head size cannot be determined from the images. As is well known, each pose has 6 degrees of freedom. Therefore the total number of unknowns is 16, and the total number of equations is 20. If we instead use their 3D coordinates as unknowns as in any typical bundle adjustment algorithms, we would end up with 20 unknowns. By using the generic properties of the face structure, the system becomes over-constrained, making the pose determination more robust.

To make the system even more robust, we add an inequality constraint on e . The idea is to force e to be positive and not too large compared to a, b, c, d . This is obvious since the nose is always out of plane π . In particular, we use the following inequality:

$$0 \leq e \leq 3a \quad (3)$$

We chose 3 as the upper bound of e/a simply because it seems reasonable to us and it works well. The inequality constraint is finally converted to equality constraint by using penalty function.

$$P_{\text{nose}} = \begin{cases} e * e & \text{if } e < 0; \\ 0 & \text{if } 0 \leq e \leq 3a; \\ (e - 3a) * (e - 3a) & \text{if } e > 3a. \end{cases} \quad (4)$$

In summary, based on equations (1), (2) and (4), we estimate $a, b, c, d, e, (\mathbf{R}, \mathbf{t})$ and $(\mathbf{R}', \mathbf{t}')$ by minimizing

$$\mathcal{F}_{5\text{pts}} = \sum_{i=1}^5 w_i (\|\mathbf{m}_i - proj(\mathbf{R}\mathbf{p}_i + \mathbf{t})\|^2 + \|\mathbf{m}'_i - proj(\mathbf{R}'\mathbf{p}_i + \mathbf{t}')\|^2) + w_n P_{\text{nose}} \quad (5)$$

where w_i 's and w_n are the weighting factors, reflecting the contribution of each term. In our case, $w_i = 1$ except for the nose term which has a weight of 0.5 because it is usually more difficult to locate the nose top than other feature points. The weight for penalty, w_n , is set to 10. The objective function (5) is minimized using a Levenberg-Marquardt method [4]. More precisely, as mentioned earlier, we set a to a constant during minimization since the global head size cannot be determined from images.

3 Incorporating image point matches

If we estimate camera motion using only the five user marked points, the result is sometimes not very accurate because the markers contain human errors. In this section, we describe how to incorporate the image point matches (obtained by any feature matching algorithm) to improve precision.

Let $(\mathbf{m}_j, \mathbf{m}'_j)$ ($j = 1, \dots, K$) be the K point matches, each corresponding to the projections of a 3D point \mathbf{p}_j according to the perspective projection (1) and (2). Obviously, we have to estimate \mathbf{p}_j 's which are unknown. Assuming that each image point is extracted with the same accuracy, we can estimate $a, b, c, d, e, (\mathbf{R}, \mathbf{t}), (\mathbf{R}', \mathbf{t}')$ and $\{\mathbf{p}_j\}$ ($j = 1, \dots, K$) by minimizing

$$\mathcal{F} = \mathcal{F}_{5\text{pts}} + w_p \sum_{j=1}^K (\|\mathbf{m}_j - proj(\mathbf{R}\mathbf{p}_j + \mathbf{t})\|^2 + \|\mathbf{m}'_j - proj(\mathbf{R}'\mathbf{p}_j + \mathbf{t}')\|^2) \quad (6)$$

where $\mathcal{F}_{5\text{pts}}$ is given by (5), and w_p is the weighting factor. We set $w_p = 1$ by assuming that the extracted points have the same accuracy as those of eye corners and mouth corners. The minimization can again be performed using a Levenberg-Marquardt method.

This is a quite large minimization problem since we need to estimate $16 + 3K$ unknowns, and therefore it is computationally quite expensive especially for large K . Fortunately, as shown in [7], we can eliminate the 3D points using a first order approximation. The following term

$$\|\mathbf{m}_j - proj(\mathbf{R}\mathbf{p}_j + \mathbf{t})\|^2 + \|\mathbf{m}'_j - proj(\mathbf{R}'\mathbf{p}_j + \mathbf{t}')\|^2$$

can be shown to be equal, under the first order approximation, to

$$\frac{(\tilde{\mathbf{m}}_j'^T \mathbf{E} \tilde{\mathbf{m}}_j)^2}{\tilde{\mathbf{m}}_j^T \mathbf{E}^T \mathbf{Z} \mathbf{Z}^T \mathbf{E} \tilde{\mathbf{m}}_j + \tilde{\mathbf{m}}_j'^T \mathbf{E}'^T \mathbf{Z}' \mathbf{Z}'^T \mathbf{E}' \tilde{\mathbf{m}}_j'}$$

where $\tilde{\mathbf{m}}_j = [\mathbf{m}_j^T, 1]^T$, $\tilde{\mathbf{m}}'_j = [\mathbf{m}'_j{}^T, 1]^T$, $\mathbf{Z} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$, and \mathbf{E} is the essential matrix to be defined below.

Let $(\mathbf{R}_r, \mathbf{t}_r)$ be the relative motion between two views. It is easy to see that

$$\begin{aligned} \mathbf{R}_r &= \mathbf{R}'\mathbf{R}^T \\ \mathbf{t}_r &= \mathbf{t}' - \mathbf{R}'\mathbf{R}^T\mathbf{t} \end{aligned}$$

Furthermore, let's define a 3×3 antisymmetric matrix $[\mathbf{t}_r]_{\times}$ such that $[\mathbf{t}_r]_{\times}\mathbf{x} = \mathbf{t}_r \times \mathbf{x}$ for any 3D vector \mathbf{x} . The essential matrix is then given by

$$\mathbf{E} = [\mathbf{t}_r]_{\times}\mathbf{R}_r \quad (7)$$

which describes the epipolar geometry between two views [1].

In summary, the objective function (6) becomes

$$\begin{aligned} \mathcal{F} &= \mathcal{F}_{5\text{pts}} \\ &+ w_p \sum_{j=1}^K \frac{(\tilde{\mathbf{m}}_j{}^T \mathbf{E} \tilde{\mathbf{m}}_j)^2}{\tilde{\mathbf{m}}_j{}^T \mathbf{E}^T \mathbf{Z} \mathbf{Z}^T \mathbf{E} \tilde{\mathbf{m}}_j + \tilde{\mathbf{m}}_j{}^T \mathbf{E} \mathbf{Z} \mathbf{Z}^T \mathbf{E}^T \tilde{\mathbf{m}}_j} \end{aligned} \quad (8)$$

Notice that this is a much smaller minimization problem. We only need to estimate 16 parameters as in the five-point problem (5), instead of $16 + 3K$ unknowns.

To obtain a good initial estimate, we first use only the five feature points to estimate the head motion by using the algorithm described in Section 2. Thus we have the following two step algorithm:

Step1. Set $w_p = 0$. Solve minimization problem 8.

Step2. Set $w_p = 1$. Use the results of step1 as the initial estimates. Solve minimization problem 8.

Notice that we can apply this idea to the more general cases where the number of feature points is not five. For example, if there are only two eye corners and mouth corners, we'll end up with 14 unknowns and $16 + 3K$ equations. Other symmetric feature points (such as the outside eye corners, nostrils, etc) can be added into equation 8 in a similar way by using local coordinate system Ω_0 .

4 Results

In this section, we show some test results to compare our new algorithm with the traditional algorithms. Since there are multiple traditional algorithms, we chose to implement the algorithm as described in [6]. It works by first computing an initial estimate of the head motion from the essential matrix [1], and then re-estimate the motion with a nonlinear least-squares technique.

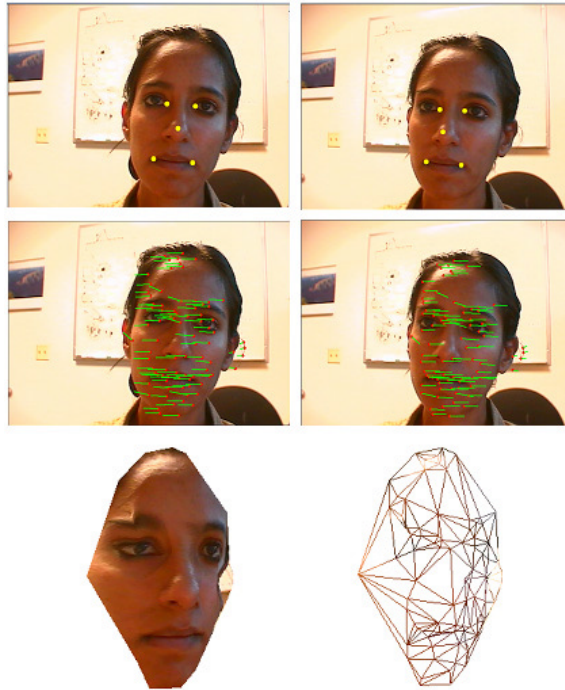


Figure 2: Top row: a pair of images with five markers. Middle row: image matching points. Bottom row: a novel view of the 3D reconstruction of the image matching points with the head motion computed by our new algorithm.

We have run both the traditional algorithm and our new algorithm on many real examples. We found many cases where the traditional algorithm fails while our new algorithm successfully results in reasonable camera motions. Figure 2 is such an example. The top row shows a pair of images with five markers each. The middle row shows the image matching points which are obtained by using a feature-based image matching algorithm. The green lines are the motion vectors of the matches. The motion computed by the traditional algorithm is completely bogus, and the 3D reconstructions give meaningless results. But our new algorithm gives a reasonable result. We generate 3D reconstructions based on the estimated motion, and perform Delauney triangulation. The left image at the bottom row shows the texture mapped triangles at a new pose (the top left image is used as the texture map), and the image on the right shows its wire frame.

In order to know the ground truth, we have also performed experiments on artificially generated data. We arbitrarily select 80 vertices from a 3D face model (Figure 4) and project its vertices on two views (the head motion is eight degrees apart). The image size is 640 by 480 pixels. We also project the five 3D feature points (eye corners, nose top, and mouth corners) to generate the image coordinates of the markers. We then add random noises to the coordinates (u, v) of both the image

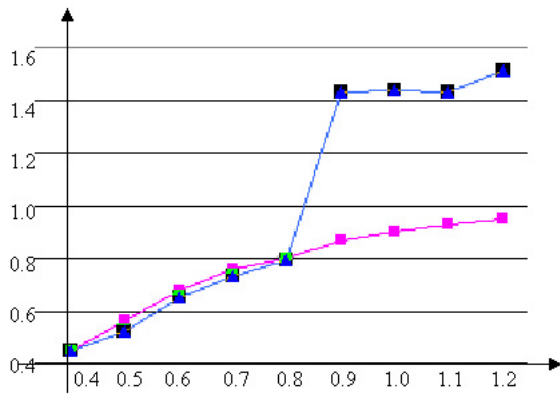


Figure 3: Comparison of the new algorithm with the traditional algorithm. The blue curve shows the results of the traditional algorithm and the red curve shows the results of our new algorithm. The horizontal axis is the noise variance. The vertical axis is the error of computed head motion.

points and the markers. The noises are generated by a pseudo-random generator subject to Gaussian distribution with zero mean and variance ranging from 0.4 to 1.2. Notice that we add noises to the marker's coordinates as well. The results are plotted in Figure 3. The blue curve shows the results of the traditional algorithm and the red curve shows the results of our new algorithm. The horizontal axis is the variance of the noise distribution. The vertical axis is the difference between the estimated motion and the actual motion. The translation vector of the estimated motion is scaled so that its magnitude is the same as the actual motion. The difference between two rotations is measured as the Euclidean distance between the two rotational matrices. We can see that as the noise increases, the error of the traditional algorithm has a sudden jump at certain point. But the errors of our new algorithm grow much more slowly.

5 Conclusion

We have developed a new head motion estimation algorithm which takes advantage of the physical properties of human face features. The algorithm significantly improves the robustness over the traditional method. It can be applied to human face modeling and tracking systems where the markers can be obtained either through user intervention or by using automatic feature detection algorithms. This algorithm can be easily extended to general cases where the number of feature points is not necessarily five.

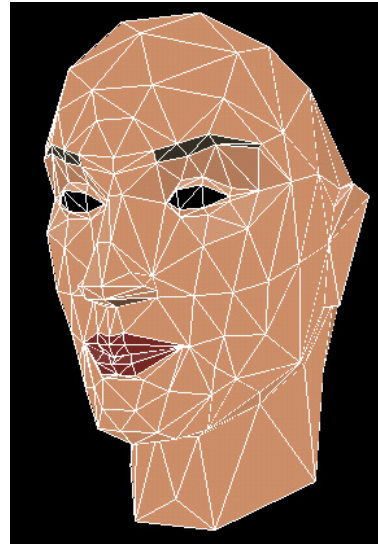


Figure 4: The face model used for experiments .

References

- [1] O. Faugeras. *Three-Dimensional Computer Vision: a Geometric Viewpoint*. MIT Press, 1993.
- [2] P. Fua. Using model-driven bundle-adjustment to model heads from raw video sequences. In *International Conference on Computer Vision*, pages 46–53, Sept. 1999.
- [3] T. S. Jebara and A. Pentland. Parameterized structure from motion for 3d adaptive feedback tracking of faces. In *Proc. CVPR*, pages 144–150, 1997.
- [4] J. More. The levenberg-marquardt algorithm, implementation and theory. In G. A. Watson, editor, *Numerical Analysis*, Lecture Notes in Mathematics 630. Springer-Verlag, 1977.
- [5] F. Pighin, J. Hecker, D. Lischinski, R. Szeliski, and D. H. Salesin. Synthesizing realistic facial expressions from photographs. In *Computer Graphics, Annual Conference Series*, pages 75–84. Siggraph, July 1998.
- [6] Z. Zhang. Motion and structure from two perspective views: From essential parameters to euclidean motion via fundamental matrix. *Journal of the Optical Society of America A*, 14(11):2938–2950, 1997.
- [7] Z. Zhang. On the optimization criteria used in two-view motion analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(7):717–729, 1998.