Liu M.L.and Wong K.H., "Pose Estimation Using Four Corresponding Points",
Pattern Recognition Letters, Volume 20, Number 1 January 1999, pp. 69-74.

Pose Estimation using Four Corresponding Points

M. L. LIU K. H. WONG

Department of Compter Science and Engineering

Chinese University of Hong Kong

Shatin, Hong Kong, China

September 11, 1998

Abstract

In this paper, a novel model-based pose estimation system is developed. In this system, correspondences between four points on a rigid solid object and four points in an image are used to estimate the new position and orientation of that rigid object under full perspective projection. Synthetic and real data have been used to verify our method.

Keyword **:** Model-based pose estimation, Gauss-Newton, Shape alignment

1 Introduction

Model-based pose estimation in computer vision is an important research problem applicable to many areas such as robot navigation, motion tracking and single camera calibration. Many methods have been proposed and discussed in (Huang and Netravali, 1994). For example, Fischler and Bolles (Fischler and Bolles, 1981) proposed

a method by using three non-collinear points. However, they revealed that all the Perspective Three-Point(P3P) problems can have as many as four possible solutions (excluding the reflection). Formation of these multiple solutions was later explained by Wolfe, Mathis, Sklair and Magee (Wolfe et al., 1991). On the other hand, Huttenlocher and Ullman (Huttenlocker and Ullman, 1988) proved that 3 non-collinear model points are enough to align a model, however, it is under weak perspective projection. Our algorithm is also similar to that in (Lowe, 1987) and (McReynolds and Lowe, 1996) but we use the length of the vectors from the focus point of the camera to the features as the variables in the Gauss-Newton method. However, geometric transformations (rotation and translation) are used instead in Lowe's algorithm. The number of unknowns in our algorithm is 4 instead of 6 in Lowe's algorithm, which is an advantage and makes our system more efficient.

Although, there are some direct and linear methods for pose estimation such as (Faugeras, 1993) and (Quan and Lan, 1998). However, Faugeras's work can only be used for the problem with 6 or more point correspondences and the model points must not be coplanar. Also, the rigidity of the object after pose estimation may not be ensured (an example is shown in Figure 1) in Quan's work.

In this paper, we are going to propose an efficient and accurate algorithm to the P4P problem under the full perspective viewing model. Fischler and Bolles (Fischler and Bolles, 1981) revealed that an unique solution cannot be assured for the P4P problems (unless the 4 points are coplanar in 3D space). However, it is obvious that the possibility for the occurrence of multiple solutions in the P4P problem is much smaller than that in the P3P problem. Therefore, using a 4-point model is much more reliable than using a 3-point model. Also, our system has been examined by using both synthetic data and real data taken by a calibrated camera (M. Magee et al., 1990) with good results.

2 Algorithm

The object being tracked and its projection on the image plane at time t are shown in Figure 2. In the figure, $\{A_{n,t}\}$ is the set of the 3D model points on the object being tracked, $\{a_{n,t}\}$ is the set of the corresponding 2D projected image points. Also $L = \{l_{n,t}\}$ is the set of the distances between model points $\{A_{n,t}\}$ and the optical center p of a calibrated camera with focal length f, while n is the index of the model points which can be any value of $1, 2, 3$ or 4.

$$\overrightarrow{PA_{n,t}} = l_{n,t}\overrightarrow{u_{n,t}} \tag{1}$$

where

$$\overrightarrow{u_{n,t}} = \frac{\overrightarrow{P(a_{n,t}, f)}}{\left|\overrightarrow{P(a_{n,t}, f)}\right|} \tag{2}$$

In which $\overrightarrow{P(a_{n,t}, f)}$ is a vector from P to the 3D point $(a_{n,t}, f)$ on the image plane, and $a_{n,t}(= (x_{n,t}, y_{n,t}))$ is the 2D coordinates of the $n-th$ image point at time t.

In our algorithm, $\left\{\left|\overrightarrow{A_{i,t}A_{j,t}}\right|\right\}_{i,j=1,2,3,4.}$ are measured manually before tracking begins. At $t = 0$, $\{a_{n,0}\}$ are picked up manually. Subsequently, the inputs $\{a_{n,1}\}$, $\{a_{n,2}\}$, ... are obtained by correlation between windows (size=5x5) extracted from two consecutive images. And the outputs of our algorithm are the new positions of the 3D model points $\{A_{n,t}\}$ which will be obtained using the Gauss-Newton method.

At any time t, apply the constraints to preserve the distances between model points, hence the following error functions are obtained:

$$e^k(i,j,t) = \left\{(l^k_{i,t})^2 + (l^k_{j,t})^2 - 2l^k_{i,t}l^k_{j,t}\,(\overrightarrow{u_{i,t}} \bullet \overrightarrow{u_{j,t}})\right\} - \left|\overrightarrow{A_{i,t}A_{j,t}}\right|^2 \tag{3}$$

where i,j=1,2,3 or 4 with $i < j$.

Therefore, there are totally six error functions in the form of equation (3) with different combinations of i and j. However, it is not enough to reconstruct the model

by these six constraints only. For example, Figure 1 shows there are two sets of model points that can also satisfy the same set of the six constraints. In order to preserve the rigidity of the object (the shape of the model) so that the model can be reconstructed uniquely, another error function is also required.

$$\begin{aligned} g^k(t) &= \left[\left(l^k_{2,t}\overrightarrow{u_{2,t}} - l^k_{3,t}\overrightarrow{u_{3,t}}\right) \times \left(l^k_{4,t}\overrightarrow{u_{4,t}} - l^k_{3,t}\overrightarrow{u_{3,t}}\right)\right] \bullet \left(l^k_{1,t}\overrightarrow{u_{1,t}} - l^k_{3,t}\overrightarrow{u_{3,t}}\right) \\ &\quad - \left(\overrightarrow{A_{3,t}A_{2,t}} \times \overrightarrow{A_{3,t}A_{4,t}}\right) \bullet \overrightarrow{A_{3,t}A_{1,t}} \end{aligned} \tag{4}$$

Assuming $\mathbf{E}$ is the vector of all $e^k(i,j,t)$ and $g^k(t)$ after the $k-th$ iteration, that is $\mathbf{E} = [e^k(1,2,t), e^k(1,3,t), e^k(1,4,t), e^k(2,3,t), e^k(2,4,t), e^k(3,4,t), g^k(t)]^T$. $\mathbf{J}$ is the Jacobian matrix, $\mathbf{J}_{mn} = \frac{\partial \mathbf{E}_m}{\partial l_n}$,where $m = 1, 2, ..$ or 7, and $n = 1, 2, 3$ or 4. Then by the Gauss-Newton method, we have the following update rules to minimize the error function:

$$\mathbf{L}^{k+1} = \mathbf{L}^k - \mathbf{h} \tag{5}$$

where $\mathbf{L}^k$ is the vector of predicted lengths $\{l_{n,t}\}_{n=1,2,3,4,}$ at iteration k,

$$\mathbf{Jh} = \mathbf{E} \tag{6}$$

and

$$\mathbf{h} = \left(\mathbf{J}^T\mathbf{J}\right)^{-1}\mathbf{J}^T\mathbf{E} \tag{7}$$

By applying equations (5) to (7) iteratively with a reasonable initial guess until $\mathbf{L}$ is stable , 3D coordinates of all the object points $\{A_{n,t}\}$ can also be obtained by Equation (1).

3 Experimental result

Both synthetic and real data are used to examine our algorithm. Those experiments are done on an SGI-INDY computer using MATLAB 4.2c.

3.1 Experiment using Synthetic Data

In the experiment using synthetic data, both Lowe's algorithm and our proposed algorithm iterate until one of the following criteria is met:-

1. Each element in the error vector is less than 10^{-3} units.

2. The number of iterations exceeds an upper bound of 300.

In the experiment, 30 different synthetic 3D objects are generated randomly for testing. Each object contains 4 feature points that are distributed in a cube of $20 \times 20 \times 20$ units with the center at $[0, 0, 120]$. A sequence of 20 images is taken for each object and the motions of the rigid objects are randomly generated. Also, except for the first image of a sequence, the iterative algorithms chooses the result of the previous image as the initial guess. There are totally 600 images for each test. In order to simulate the image taken by a digital camera, the projection of the points is digitized by rounding off to integer pixel coordinates.

Except for the first image, the system chooses the result of the previous image as the initial guess for both iterative algorithms. It is reasonable because the translation and rotation between two consecutive frames in a realistic movie should be small. Also, there are upper bounds to govern the variations of the rotation and the translation. The experimental result is shown in Table (1)-(6).

By comparing the result shown in Table (2) with that in Table (4), it is shown that our algorithm has a better performance than Lowe's algorithm since the mean value

algorithm	min. iter.	max. iter.	Mean (iter.)	Time (s)
Lowe's	3	11	3.700	32
Our	2	10	3.252	13

***Note:- the term 'iter.' indicates the number of iterations for the systems to converge to a stable answer. Also, 's' stands for 'second'.

Table 1: Result for synthetic data:- bound of rotational angle = 9^o and bound of translation in each axis = 10 units

algorithm	min. iter.	max. iter.	Mean (iter.)	Time (s)
Lowe's	3	22	4.383	36
Our	2	42	3.220	13

Table 2: Result for synthetic data:- bound of rotational angle = 18^o and bound of translation in each axis = 10 units

of the iteration required in our algorithm is much less than that in Lowe's algorithm. The reason for this phenomenon is that, the distance between the model points and the center of perspective projection remains unchanged between two consecutive images or frames if the motion between them is pure rotation. This effect can also be found by comparing the result shown in Table (1) and Table (3).

Also, by Table (2), (4) and (6), it is shown that our algorithm declines as the bound of the translation (in each axis) increases. Relatively, the importance of the rotational motion becomes smaller. In the extreme case, the result in Table (5) shows that the performance of our algorithm is much worse than Lowe's algorithm (in terms of the mean values of the iterations) when the motion is pure translational.

As a result, we can conclude that the performance of our algorithm is better in the cases that the transformation of the object is dominated by the rotational motion. Therefore, our algorithm is suitable for the environment where the motion of the rigid

algorithm	min. iter.	max. iter.	Mean (iter.)	Time (s)
Lowe's	3	6	3.780	31
Our	1	9	1.8167	11

Table 3: Result for synthetic data:- bound of rotational angle = 9^o and bound of translation in each axis = 0 unit

algorithm	min. iter.	max. iter.	Mean (iter.)	Time (s)
Lowe's	3	51	4.518	37
Our	1	11	1.747	11

Table 4: Result for synthetic data:- bound of rotational angle = 18^o and bound of translation in each axis = 0 unit

algorithm	min. iter.	max. iter.	Mean (iter.)	Time (s)
Lowe's	2	7	2.075	19
Our	3	84	4.025	15

Table 5: Result for synthetic data:- bound of rotational angle = 0^o and bound of translation in each axis = 40 units

algorithm	min. iter.	max. iter.	Mean (iter.)	Time (s)
Lowe's	3	13	4.310	35
Our	3	120	4.172	15

Table 6: Result for synthetic data:- bound of rotational angle = 18^o and bound of translation in each axis = 40 units

object is dominated by rotation.

On the other hand, the computation time for our algorithm is much less than that for Lowe's algorithm. It is because the number of the parameters (or unknown to be found) in Lowe's algorithm is 6 which is more than that required by our algorithm. Since, both these two algorithms use the Gauss-Newton method (Athinson, 1989), in which expensive calculations about the multiplication and inversion of matrices (especially for the Jacobian matrix) are extensively used. Therefore, the smaller the size of the matrices (i.e. the error vector and especially for the Jacobian matrix), the shorter the computation time. As a result, the computational time for our proposed algorithm is much shorter.

3.1.1 Experiment using Real Data

In the experiment using real data, images of an input device is taken by a calibrated camera (M. Magee et al., 1990). This device is made of a black card and eight LEDs as shown in Figure 3. The four LEDs at the outermost layer (i.e. the LEDs marked by the circles in Figure 3) are used as the four model points. A short movie is created by capturing the random motion of this device. Then the correspondences between the four model points (LEDs) and the image points in each frame of the movie are found by using the cross-correlation technique. This movie contains 113 frames. By using our algorithm, about 28 frames can be handled in each second. In this experiment, the limit for the stopping criterion is set to be 10^{-3}cm.

Actually, the object shown in Figure 3 is used to illustrate that our proposed algorithm is applicable to the case when all the four model points are lying on the same plane. For the case when the model points are not coplanar (i.e. not all the points lying on the same plane), the object in Figure 4 is used.

A box is shown in Figure 4, in which the four corners marked by circles are used

as the model points. A sequence of images of the box is captured. By applying our algorithm, the pose of this object is estimated. Figure 5 and 6 indicate two samples of the video sequence. In these two figures, the black lines are used to indicate the estimated projection of the result of the pose estimation of this object. By using these two figures, we show that the accuracy of our algorithm is also good when the four model points are not on the same plane. Also, the speed of the computation is about 28 frames in one second.

From this experiment, it is shown that our algorithm is very fast and can be applicable to both cases when either all the model points are coplanar or not.

4 Conclusion and Discussion

In this paper, a novel iterative solution for the pose estimation problem using 4 model points is introduced. Actually, this algorithm may be extended to handle PnP problems with $n > 4$. However, the number of error functions may increase rapidly with the increase in the number of points used (the total number of Equation (3) should be ${}_nC_2$ and number of Equation (4) should be ${}_nC_4$). Also, for the case of 6 or more points and not all the model points are lying on the same plane, linear and direct methods can be used (Faugeras, 1993). We are currently working on such an efficient algorithm for the P4P problem.

On the other hand, a visual tracking system is developed and examined by both synthetic and real data. The result suggests that this technique can be used to develop a 6D (including translation and rotation parameters) vision based mouse.

References

Athinson, K. (1989). *An Introduction to Numerical Analysis*. John Wiley and Sons, 2nd. edition.

Faugeras, O. (1993). *Three-Dimensional Computer Vision*. MIT Press.

Fischler, M. A. and Bolles, R. (1981). Random sample consensus:- a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395.

Huang, T. S. and Netravali, A. N. (1994). Motion and structure from feature correspondences:- a review. *Proceeding of The IEEE*, 82(2):252–268.

Huttenlocker, D. P. and Ullman, S. (1988). Recognizing solid objects by alignment. *presented at the DARPA Image Understanding Workshop, Cambridge MA*, pages 1114–1122.

Lowe, D. (1987). Three-dimensional object recognition form single two-dimensional images. *Artificial Intellengence*, 31:355–395.

M. Magee, W. H., Gatrell, L., Marietta, M., and Wolfe, W. (1990). Adaptive camera calibration in an industrial robotic environment. *IEA/AIE 90*, vol. I:242–251.

McReynolds, D. and Lowe, D. (1996). Rigidity checking of 3d point correspondences under perspective projection. *IEEE Trans. on Pattern Analysis and Machine Intell.*, 18(12):1174–1185.

Quan, L. and Lan, Z. (1998). Linear n¿=4 -point pose determination. *ICCV'98*.

Wolfe, W., Mathis, D., Sklair, C., and Magee, M. (1991). The perspective view of three points. *IEEE Transactions on Pattern Analysis and Machine Intell.*, 13(1):66–73.

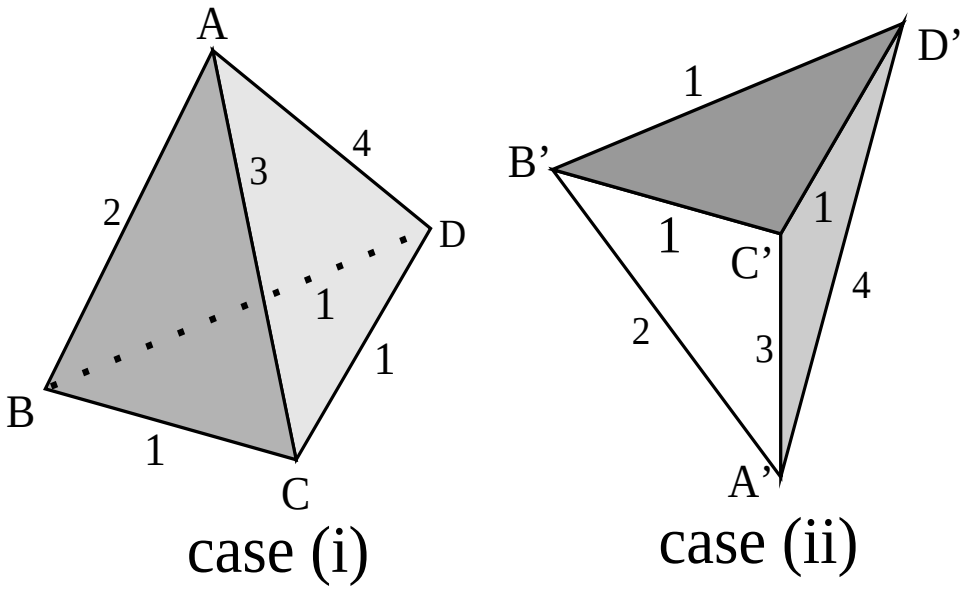

Figure 1: Two models with the same sides (AB=A’B’, BC=B’C’ and etc...), but with different orientations (i.e. different shapes). In case (i), A is above the plane formed by BCD, while in case (ii), it is under that plane.

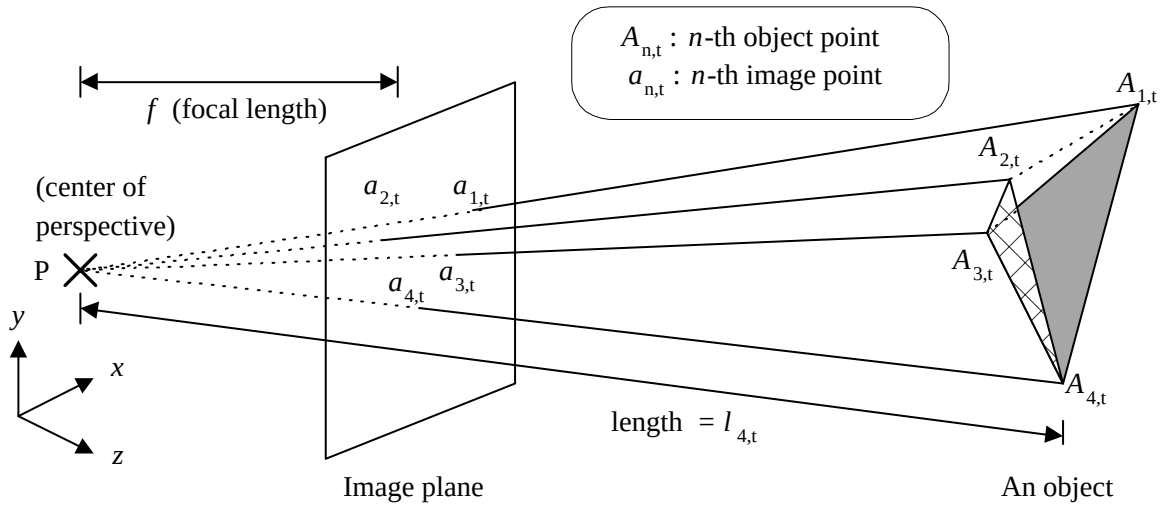

Figure 2: A model with 4 model points and their corresponding image points at time t

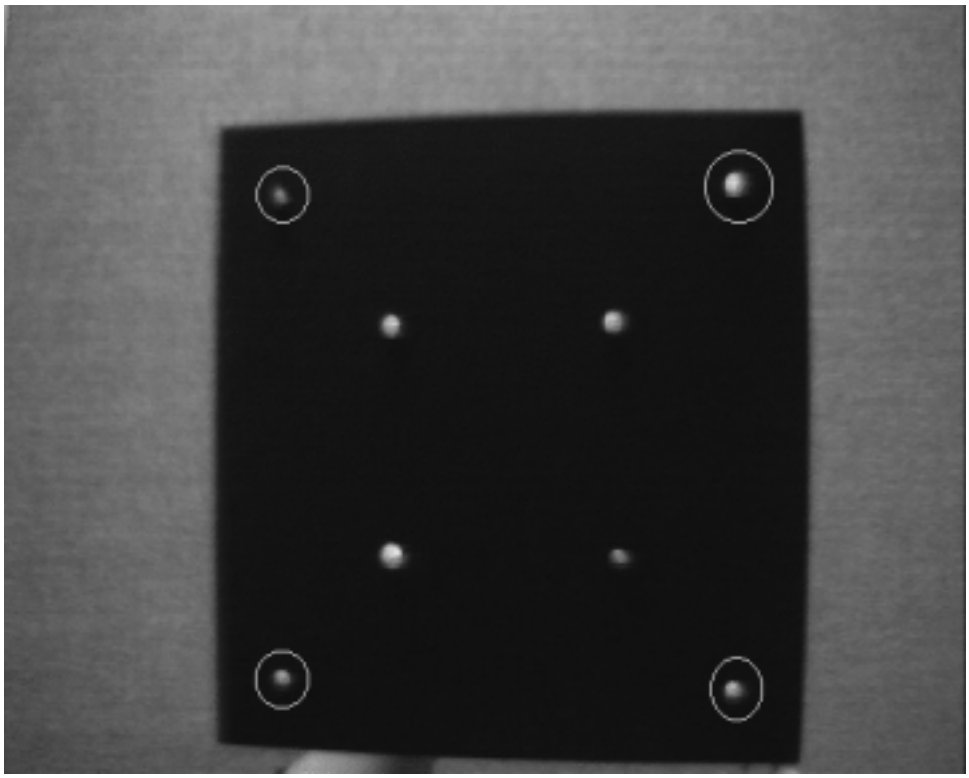

Figure 3: The input device. The four LEDs inside circles are used as 4 model points

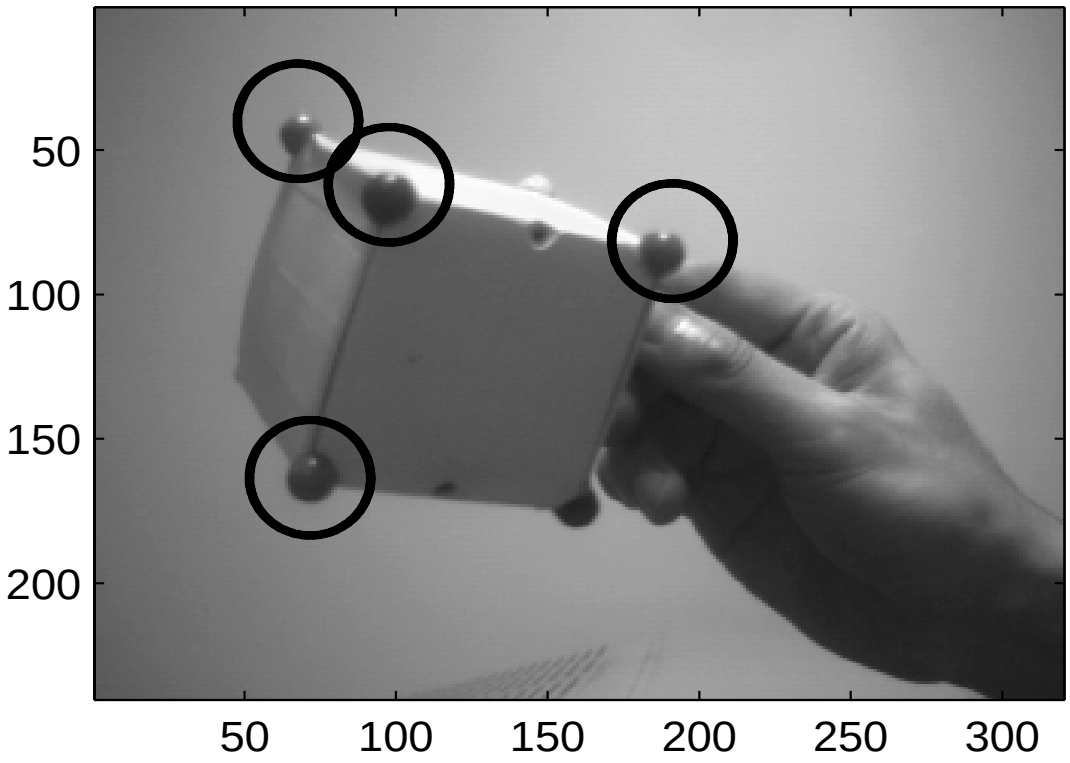

Figure 4: The second input device. The four corners inside circles are used as 4 model points

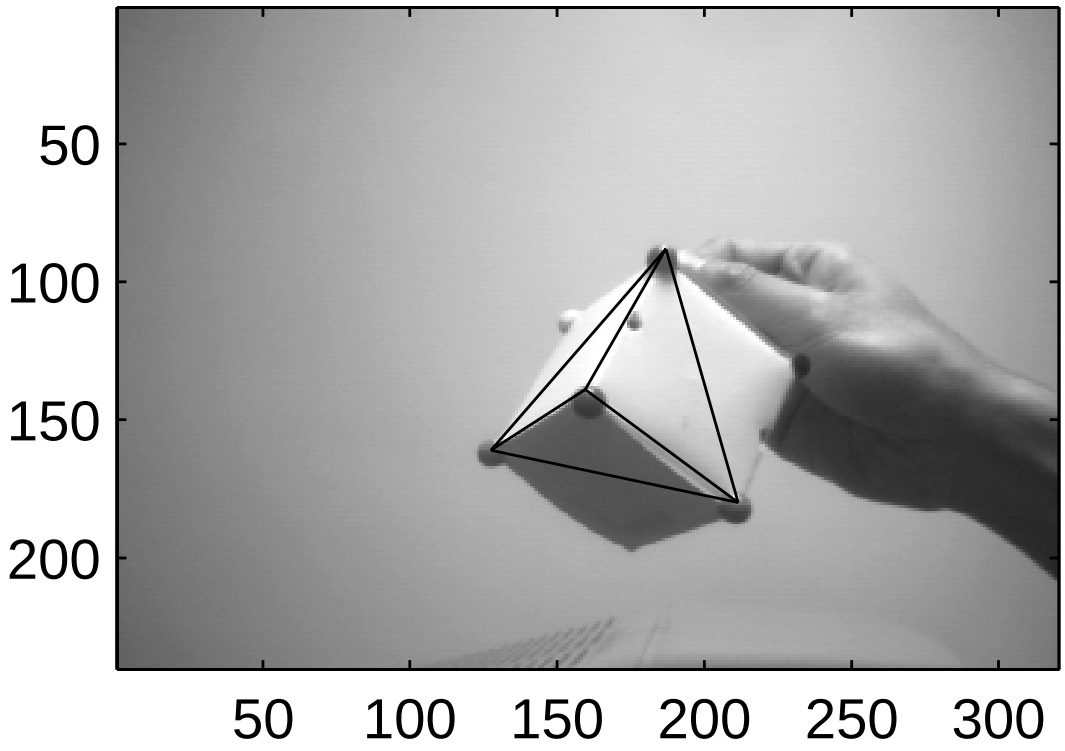

Figure 5: The sample of the result of the 1st image from the image sequence.

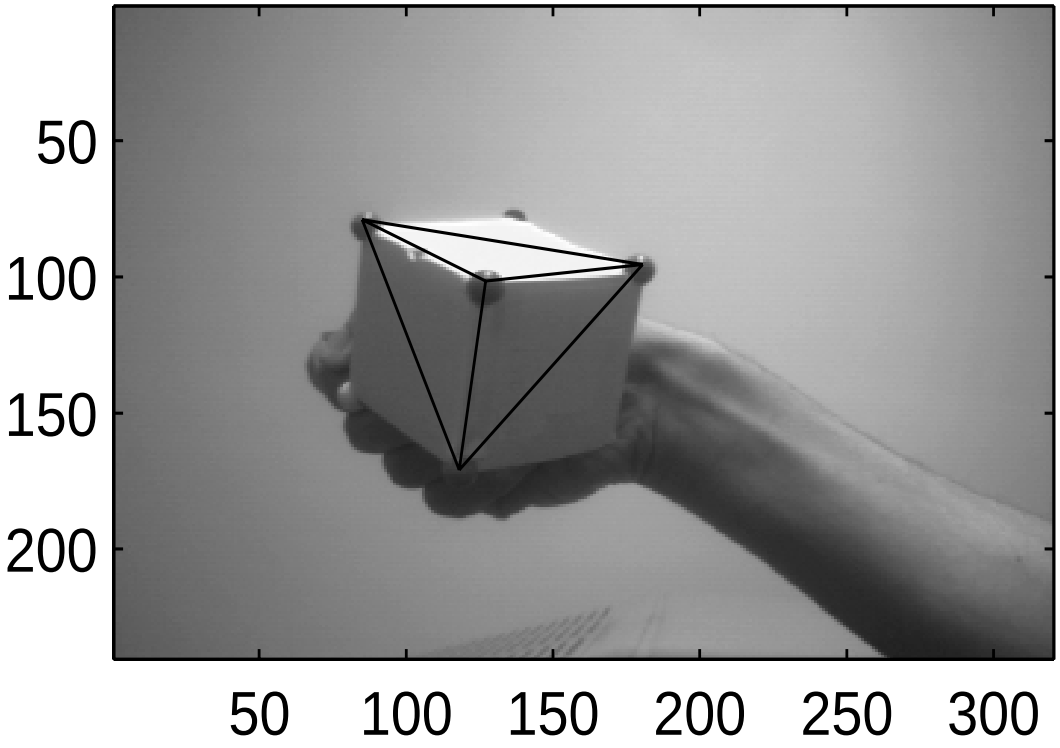

Figure 6: The sample of the result of the 2nd image from the image sequence.