

Pose Estimation for Augmented Reality Applications Using Genetic Algorithm

Ying Kin Yu*, Kin Hong Wong and Michael Ming Yuen Chang

Abstract

This paper describes a genetic algorithm that tackles the pose estimation problem in the field of computer vision. Our genetic algorithm can find the rotation and translation of an object accurately when the 3D structure of the object is given. In our implementation, each chromosome encodes both the pose and the indexes to the selected point features of the object. Instead of only searching for the pose of the object as in many of the existing work, our algorithm at the same time searches for a set containing the most reliable feature points in the process. This mismatch filtering strategy successfully makes the algorithm more robust under the presence of point mismatches and outliers in the images. Our algorithm has been tested with both synthetic and real data with good results. The accuracy of the recovered pose is compared to the existing algorithms. Our approach outperformed the Lowe's method and the other two genetic algorithms under the presence of point mismatches and outliers. In addition, our algorithm has been used to estimate the pose of a real object, which is applicable to augmented reality applications. For example, the pose obtained was used for inserting artificial objects into an augmented reality movie in this paper.

Index Terms

Pose Estimation, Genetic Algorithms, Augmented Reality.

I. INTRODUCTION

THE research work presented in this paper falls into the category of pose estimation in the field of computer vision. The goal is to estimate the pose (Roll, Pitch, Yaw rotation angles and the t_x , t_y , t_z translation parameters) of an object in the 3D space given its structure. Pose estimation is important

Ying Kin Yu is with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, Hong Kong. Email: ykyu@cse.cuhk.edu.hk Phone:+852-26098438 Fax:+852-26035024

Kin Hong Wong is with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, Hong Kong. Email: khwong@cse.cuhk.edu.hk Phone:+852-26098397 Fax:+852-26035024

Michael Ming Yuen Chang is with the Department of Information Engineering, The Chinese University of Hong Kong, Shatin, Hong Kong. Email: mchang@ie.cuhk.edu.hk Phone:+852-26098347 Fax:+852-26035032

because it is one of the two essential steps in the alternative bundle adjustment scheme for the structure from motion problem [17]. With a better pose estimation algorithm, the recovered structure can reach a higher accuracy. The pose estimation technique is also crucial in producing augmented reality movies, in which the artificial objects can be mixed with the images of the real scene. Currently, there are two major approaches in computing the pose for augmented reality. One is the marker-based technique [15] [25], in which the scene for processing must contain a special pattern for pose estimation and camera calibration. Another is the vision-based technique [16] [24], in which no special pattern is required to compute the pose. This approach is more general and can be applied to most natural scenes. Here we present an approach that is suitable for the latter problem.

A. Previous work

There are various techniques to deal with the vision-based pose estimation problem. Early work makes use of smaller number of point features in the scene. Fishler and Bolles [18] took three feature points with the "Random Sample Consensus" method to compute the pose of an object. Horaud, Conio and Leboulleux [19] estimated the pose of an object using four non-coplanar points. The solution is computed by solving biquadratic polynomial equations of one unknown with geometric constraints. Another four-point algorithm was proposed by Liu and Wong [26].

The iterative method [23] [27] [6] is also a common approach to solve the pose estimation problem. It is an iterative steepest descent method which computes the best pose to fit the image data. The residual error between the predicted and real image positions of the point features are used to form a better prediction in the next iteration by a nonlinear optimization scheme such as Newton's method. The algorithm is executed iteratively until the residual error is finally minimized.

Another approach is to tackle the problem of pose tracking by Kalman filtering. In short, Kalman filter is an estimator for the linear-quadratic-gaussian problem. It is a problem of estimating the instantaneous state of a linear dynamic system perturbed by Gaussian white noise by using measurements linearly related to the state, but corrupted by Gaussian white noise [9]. Lippiello, Siciliano and Villaniworks [12] [13] used extended Kalman filter for motion estimation. This work finds the pose of the object based on a known CAD model from stereo images. The position and orientation of the camera are recovered in realtime and the results are applied to visual servoing of robot manipulators. Wong, Or and Chang [8] applied the CONDESATION framework to track the pose of an object for the construction of a virtual walk-through environment.

Genetic algorithm is an alternative to the traditional solutions. Most of the existing methods [2] [4] [5] model the problem as a camera calibration procedure. Hati and Sengupta [2] used the genetic algorithm framework to estimate the extrinsic parameters of a camera. Real number representation is adopted to encode the translation and rotation parameters in the chromosomes. Gaussian mutator and Blend crossover are used as the genetic operators in their genetic algorithm. The work by Ji and Zhang [4] is quite similar to [2] but the authors applied a genetic algorithm to search for both the intrinsic and extrinsic parameters of a camera. They incorporated a time dependent function for the adjustment of the step size into the mutation operator to facilitate the convergence of the fitness value. Cerveri, Pedotti and Borghese described an approach in [5] that calibrates a stereo camera system with the enhanced evolutionary search. The only work that directly addresses the pose estimation problem with genetic algorithm is by Toyama, Shoji and Miyamochi [3]. The inputs to their algorithm are the edge images instead of point features. They adopted the phenotypic forking genetic algorithm [10] to perform the search. This strategy combines the advantages of conventional genetic algorithm and steepest descent method such that the solution can be found within a smaller number of generations.

B. Our contributions

The algorithm proposed in this paper is based on the work by Hati and Sengupta [2]. We also use real numbers to represent pose in the chromosomes. Our improvement is on incorporating a feature searching strategy into our algorithm. That means while searching for the pose of an object, our algorithm also searches for the set containing the most reliable features among all the available model points in the process. To achieve this, indexes of the reliable feature points are encoded in the chromosomes. The chromosome now comprises two sections: 1) One section encodes the translation and rotation parameters 2) The other section encodes the indexes to the selected point features. The genetic operators are also modified to cope with this new encoding scheme. It is shown in the experiment that our new algorithm is less susceptible to the presence of point mismatches and outliers than the original version by Hati and Sengupta. Such a robustness has also been demonstrated in the experiment in which a virtual chair was successfully put onto the rotating turntable in the real scene.

The major advantage of our genetic algorithm is that it avoids locking into a local optimum, which is a common problem in traditional steepest descent methods. As reported in the literature [4], there exist several local extrema in the landscape of the three rotation parameters when the image measurements are

used as the objective function. Our genetic search can calculate a better pose than Lowe's method in a theoretical sense.

C. Organization of the paper

The rest of this paper is organized as follows. The modeling of the pose estimation problem is first introduced in Section II. In section III, the design and implementation of our genetic algorithm are described. In section IV, the results of the synthetic and real image experiments are demonstrated. In the first experiment, the settings of genetic algorithm parameters, together with the robustness of our algorithm to the presence of point mismatches and outliers are analyzed. An empirical comparison among our approach, the traditional Lowe's method, the genetic algorithm by Hati and Sengupta and a conventional genetic algorithm plus a RANSAC robust estimator is made. In the second experiment, the accuracy of the recovered pose is demonstrated by producing an augmented reality video, in which a virtual chair is put into the real scene. In section V, possible improvements and future work are discussed.

II. PROBLEM MODELLING

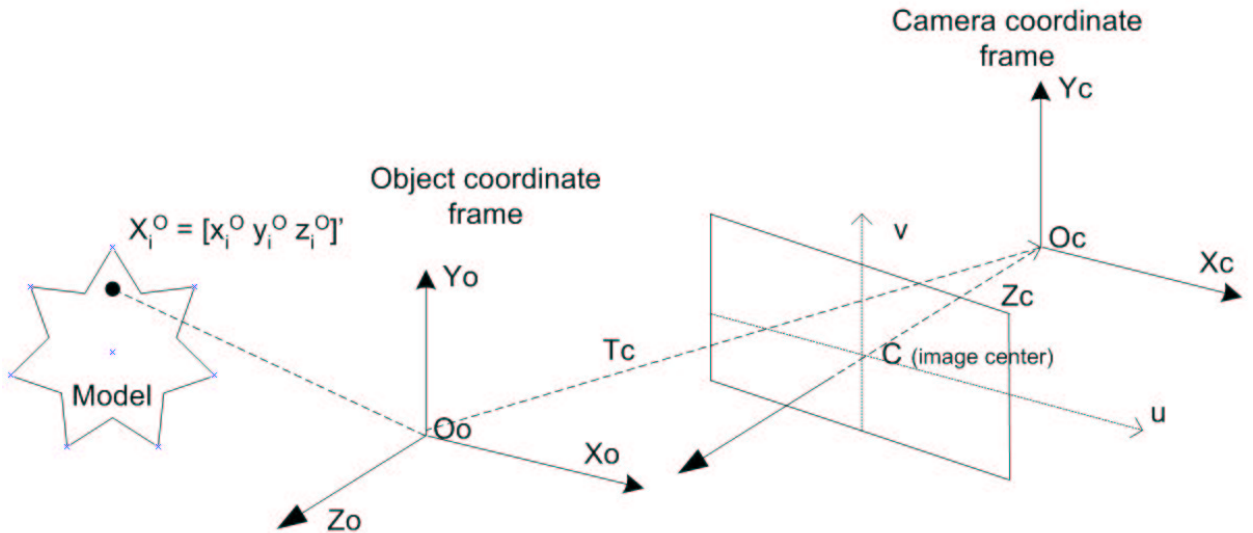


Fig. 1. The geometric model used in the paper.

Figure 1 describes the geometry of our system. $X_i^O = [x_i^O, y_i^O, z_i^O]^T$ denotes the coordinates of the point X_i with respect to the object coordinate frame. The notation $X_i^C = [x_i^C, y_i^C, z_i^C]^T$ represents the coordinates of point X_i in the camera coordinate frame. A point on the image plane is denoted by $p_i = [u_i, v_i]^T$. The

object for pose estimation is centered at the origin O_o of the object coordinate frame. The relationship between the object frame and the camera frame can be described by the following equation:

$$X_i^C = (RX_i^O + T) + T_C \quad (1)$$

R is a 3×3 rotation matrix and T is a 3×1 translation matrix. Both R and T are recovered with reference to the object frame in the process of pose estimation. T_C is a 3×1 translation matrix that brings the object in the object coordinate frame to the camera coordinate frame. It is a constant in the pose estimation process.

For the rotation matrix R , a parameterization called the rotation around the coordinate axes is adopted. The entries of R can be written as:

$$R(\alpha, \beta, \gamma) = \begin{pmatrix} \cos \beta \cos \gamma & -\cos \beta \sin \gamma & \sin \beta \\ \sin \alpha \cos \beta \cos \gamma + \cos \alpha \sin \gamma & -\sin \alpha \sin \beta \cos \gamma + \cos \alpha \sin \gamma & -\sin \alpha \cos \beta \\ -\cos \alpha \sin \beta \cos \gamma + \cos \alpha \sin \beta & \cos \alpha \sin \beta \sin \gamma + \sin \alpha \cos \gamma & \cos \alpha \cos \gamma \end{pmatrix} \quad (2)$$

The camera used in the system is calibrated and its focal length is assumed fixed. The camera model is full perspective and the projection can be mathematically represented as:

$$\begin{pmatrix} u_i \\ v_i \end{pmatrix} = \frac{f}{z_i^C} \begin{pmatrix} x_i^C \\ y_i^C \end{pmatrix} \quad (3)$$

where f is the focal length of the camera. The problem of pose estimation is to compute the rotation angles α, β, γ around each axis and the translation parameters t_x, t_y and t_z of the object for the 2D image in each time-step. In the computation, the back-projection error (BE), which is defined as:

$$BE = \sum_{j=1}^N d_j^2 \quad (4)$$

is to be optimized. In the equation, d_j is the difference in distance between a real image point and its corresponding back-projected point given the computed pose.

III. THE GENETIC ALGORITHM IMPLEMENTATION

A. Overview of the algorithm

Our implementation of the genetic algorithm follows the framework of the conventional genetic methodology as described in [1]. Starting with an initial pool of population produced randomly, the chromosomes

in the current population have a certain chance to reproduce their offsprings. Fitter chromosomes have a higher chance to be selected for reproduction using the roulette wheel proportionate selection. An offspring is reproduced either by mutation of one chromosome or crossover between two chromosomes. We adopt the concept of overlapping population in creating the next generation of chromosomes. The population of the next generation consists of a certain portion of the chromosomes from the parent generation and some from their offsprings. The proportion between two types of chromosomes in the new generation is defined by the probability of replacement. The algorithm stops until the fitness of the best chromosome converges to a desired value. The overview of our genetic algorithm can be outlined as follows.

- 1) Generate a random population consisting of n chromosomes.
- 2) Calculate the fitness value of each chromosome.
- 3) Choose the parents from the current population using the roulette wheel proportionate selection for reproduction.
- 4) Create a temporary population of offsprings by the mutation or crossover of the parents according to the corresponding probabilities.
- 5) Select, with the roulette wheel proportionate selection, the chromosomes into the next generation from the pool of the offsprings and current generation according to the probability of replacement.
- 6) Repeat Step 1) to Step 5) until one or more of the following conditions has been reached: i) The fitness of the best chromosome has reached a desired value. ii) It has no further improvements. iii) The time limit exceeds.

B. Chromosome encoding

The chromosomes of our genetic algorithm consists of $n + 6$ elements, where n is the number of selected point features. The chromosome vector q_i^t is defined as:

$$\begin{aligned}
 q_i^t &= (p_{1i}^t, p_{2i}^t) \\
 p_{1i}^t &= (\alpha, \beta, \gamma, t_x, t_y, t_z) \\
 p_{2i}^t &= (k_1, k_2, \dots, k_n)
 \end{aligned} \tag{5}$$

The notation q_i^t refers to an individual chromosome at the t^{th} generation. It is actually a composite chromosome that is a fusion of two different types of chromosomes. p_{1i}^t encodes the pose of an object. Its elements comprise the first section of the chromosome. The encoding of this section follows exactly

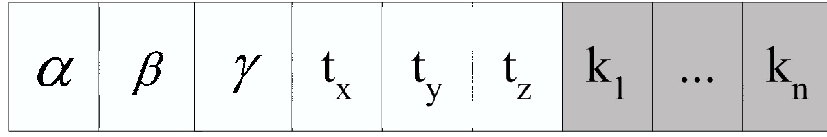


Fig. 2. An illustration of the structure of the composite chromosome.

the work of Hati and Sengupta described in [2]. Real number representation, together with an appropriate upper and lower, are adopted for each field in p_{1i}^t . The term p_{2i}^t encodes the indexes of the feature points selected for pose estimation. Its elements comprise the second section of the chromosome. Each field in p_{2i}^t stores the index to the selected model feature. Integer representation is adopted. The value n in our implementation is equal to 20, which is chosen experimentally as described in section IV A 1). Intuitively, our search attempts to find (select), using the genetic algorithm methodology, a set containing the 20 most reliable point features in the search and relies on them for pose estimation.

C. The genetic operators

1) *Mutation*: The mutation operation of the first section p_{1i}^t and the second section p_{2i}^t of the chromosomes are defined differently due to their differences in their physical meanings. For the first section p_{1i}^t , a Gaussian mutator, similar to that in [2], is adopted.

$$p_{1i}^{t+1} = p_{1i}^t + U(\lambda_1^t)S(t) \quad (6)$$

$U(\lambda_1^t)$ is a Gaussian function which generates values with Gaussian distribution according to λ_1^t , which is the variance of p_{1i}^t . $S(t)$ is a time dependent function that controls the step size of the mutation. It is defined as:

$$S(t) = 1 - r \times \exp\left(1 - \frac{t}{r'T}\right) \quad (7)$$

r and r' are two real constants that respectively lie in the interval $[0, 1]$ and $[1, 1.5]$. T is the expected total number of iterations. With that, the convergence time of the genetic algorithm is shortened. The mutation of the second section p_{2i}^t is trivial. In this section, the genes of the parent are copied to their offspring. The content of this section is preserved in the mutation operation.

$$p_{2i}^{t+1} = p_{2i}^t \quad (8)$$

2) *Crossover*: Similar to the mutation operation, the crossover operation of the first section p_{1i}^t and the second section p_{2i}^t of the chromosomes are different. For the first section, the Blend crossover [1] is adopted. In Blend crossover, a new value between two chromosomes is generated based on the interval between the two parents. The offspring has the properties inherited from both of its parents. The Blend crossover can be expressed mathematically as:

$$p_{1i}^{t+1} = Ip_{1i}^t + (1 - I)p_{1j}^t \quad (9)$$

I is a real number that ranges within $[0,1]$. For the second section p_{2i}^t of the chromosome, the crossover is done by combining two sections of chromosomes of the parents. To be more precise, the operation can be expressed mathematically as:

$$\begin{aligned} p_{2i}^{t+1} &= (k_{i1}, k_{i2}, \dots, k_{ih}, k_{j(h+1)}, k_{j(h+2)}, \dots, k_{jn}) \\ p_{2i}^t &= (k_{i1}, k_{i2}, \dots, k_{in}) \\ p_{2j}^t &= (k_{j1}, k_{j2}, \dots, k_{jn}) \end{aligned} \quad (10)$$

h is an integer that ranges within $[1,n]$. This random number h defines the point of crossover of the two parent chromosomes.

D. Fitness evaluation

To evaluate the fitness of each chromosome, the first section p_{1i}^t is decoded to get the pose represented by it. Then the second part p_{2i}^t is decoded to get the 3D coordinates of the selected model features. These selected model features are back-projected to the image plane with the pose decoded from p_{1i}^t to calculate the back-projection error (BE) as defined in equation (4). It is reasonable to use only the selected point features to evaluate the fitness function. The point features that are not selected are probably outliers in the image and may lead to a local optimum in the search. At the same time, the initial values of the second section of the composite chromosome are random combination of all available point features in the model and thus the whole set of point features are fully utilized.

Since it is originally a minimization problem, the fitness calculation is needed to be changed. The fitness value of the chromosome i is equal to:

$$f_i(BE) = C - BE \quad (11)$$

C is a constant chosen to be 10000. The problem is reformulated to be a maximization problem as in the conventional genetic search problem.

E. The roulette wheel proportionate selection

The roulette wheel proportionate selection [1] is adopted for selecting the chromosomes from the current population to the mating pool for reproduction. The concept of roulette wheel proportionate selection is simple. The chromosomes that have higher fitness values will have a higher probability to be selected. It simulates natural selection in the real world.

To perform roulette wheel proportionate selection in the implementation, we need to further scale the fitness function to suit the purpose. This is achieved by applying a sigma-truncated scaling to the fitness value in equation (11). Mathematically, sigma truncation is defined as:

$$f'_i = f_i - (\bar{f} - C'\sigma) \quad (12)$$

$$\bar{f} = \frac{1}{n} \sum_{j=1}^n f_j \quad (13)$$

f'_i is the scaled fitness. \bar{f} is the mean fitness of all the chromosomes in the population. σ is the standard deviation of the fitness of all chromosomes. C' is a real constant which can be chosen such that $C'\sigma$ is a reasonable multiple of the population's standard deviation. C' is equal to 3 in our implementation. If f'_i is negative, it is truncated to zero. With the scaled fitness, then the probability of choosing the i^{th} chromosome for reproduction is:

$$\frac{f'_i}{f'_1 + f'_2 + \dots + f'_n} \quad (14)$$

IV. EXPERIMENTS AND RESULTS

A. Experiments with synthetic data

1) *The genetic algorithm parameters:* The table in figure 3 summarizes the settings of the algorithm parameters used in the experiment. The population size and the number of selected point features were determined experimentally as follows. A synthetic object with 100 random feature points in 3D space within a cube of volume of $0.13m^3$, centered at a place 0.33m away from the viewing camera, was first generated. The camera has a focal length of 6mm. The sensor of the camera is assumed to be not perfect. It imposes a 2D zero mean Gaussian noise with standard deviation 0.1 pixels on the image captured. Since

<i>Parameters</i>	<i>Values</i>
Population size	150
Probability of mutation	0.4
Probability of crossover	0.6
Probability of replacement	0.7
Number of selected point features	20
Number of chromosome generations	130
Search range of [Yaw Pitch Roll]	0-5 degrees
Search range of $[t_x t_y t_z]$	0-5 millimeters

Fig. 3. A table showing the genetic algorithm parameters

the point correspondences between the 3D model and the 2D image may not be correct, the resulting image was made to contain 30 percents of outliers, which is assumed to be the worst condition in real cases. The object was placed randomly with a pose from $[0, 0, 0]$ to $[1, 1, 1]$ degrees for [Yaw Pitch Row] and $[0, 0, 0]$ to $[0.0005, 0.0005, 0.0005]$ meters for $[t_x, t_y, t_z]$ with reference to the camera respectively. Initially, each chromosome contains a randomly generated pose vector within the parameter bounds and a random combination of selected feature points.

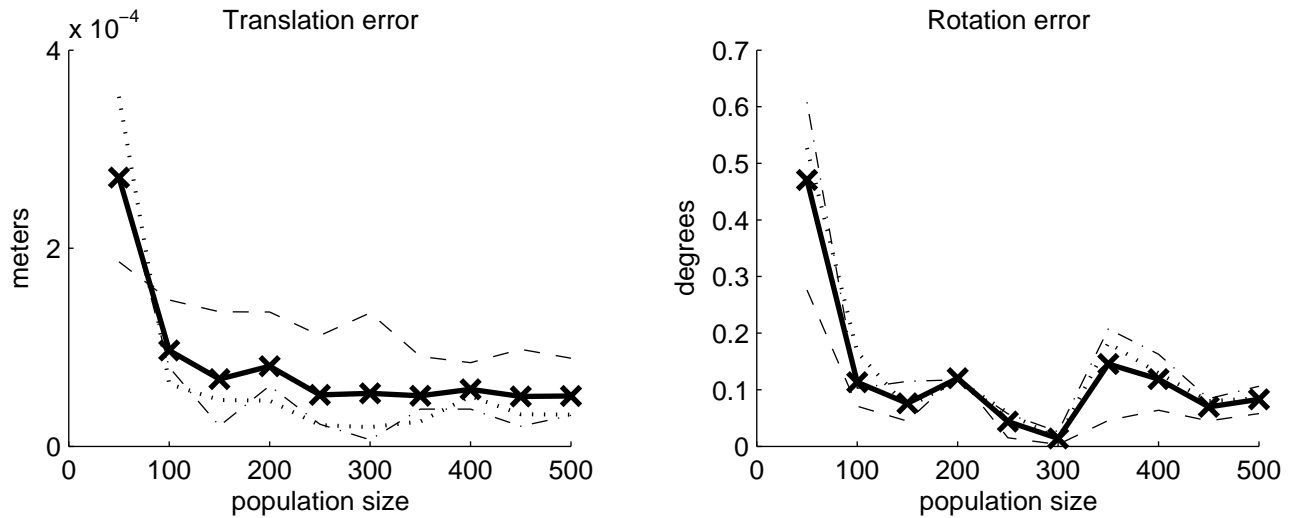


Fig. 4. A figure showing the errors of the rotation and translation parameters versus the population size. The plot on the left illustrates the translation parameter errors. The bolded solid line is the average error of the 3 translation parameters. The dash dotted line, dotted line and the dash line correspond to t_x , t_y and t_z respectively. The plot on the right illustrates the rotation parameter errors. The bolded solid line is the average error of the 3 rotation angles. The dash dotted line, dotted line and the dash line correspond to the Yaw, Pitch and Roll angle respectively.

The proposed algorithm was applied to estimate the pose of the synthetic object with different population sizes and numbers of selected point features. Figures 4 and 5 show the results, which are the averages from 50 independent data sets. From figure 4, it is found that both the rotation and translation error drop significantly when the population size increases from 50 to 150. The improvement on the accuracy becomes

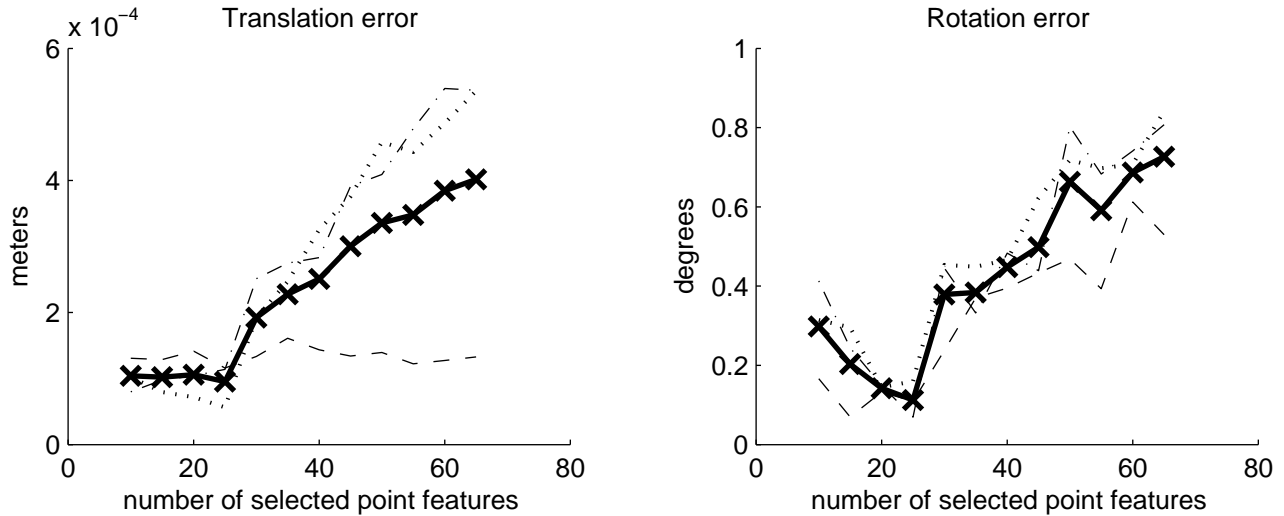


Fig. 5. A figure showing the errors of the rotation and translation parameters versus the number of selected point features. The plot on the left illustrates the translation parameter errors. The bolded solid line is the average error of the 3 translation parameters. The dash dotted line, dotted line and the dash line correspond to t_x , t_y and t_z respectively. The plot on the right illustrates the rotation parameter errors. The bolded solid line is the average error of the 3 rotation angles. The dash dotted line, dotted line and the dash line correspond to the Yaw, Pitch and Roll angle respectively.

steady when the population size is larger than 150. Since the population size is inversely proportional to the algorithm speed, the optimal size should be the minimum one that can cause the greatest reduction in error, i.e. 150, in the graph. From figure 5, you can see that the errors are increasing for the number of selected point features greater than 20. One reason for the deterioration of algorithm's convergence is that perturbed point features may increase the difficulty in searching for a fit set of parameter in such a large solution space, which is actually growing as the number of selected point features increases. This result more or less coincides with the one discussed in literature [4]. So, the optimal value 20 is chosen.

The convergence of the pose parameters of our algorithm was also investigated, which is shown in figure 6. The translation errors converge at about the 75th generation while the rotation errors falls to a steady value at round the 125th generation. The former one converges earlier since the t_x and t_y parameter are more observable from the image measurements and thus easier to be optimized. Overall, our proposed algorithm converges at the 130th generation.

Regarding the speed performance of the proposed algorithm, it takes about 0.15 seconds to compute and evaluate a generation of chromosomes, given that the proposed algorithm is implemented in Matlab and run on a Pentium 2GHz machine. Assume we cut off at the 130th generation, it takes about 19.5 seconds to estimate the pose of an image frame. Since the algorithm involves quite a lot of loops, the speed could be at least double if it is implemented in C language. Moreover, the algorithm can be further parallelized. Due to the fact that genetic algorithm is a group search, the computation of each chromosome

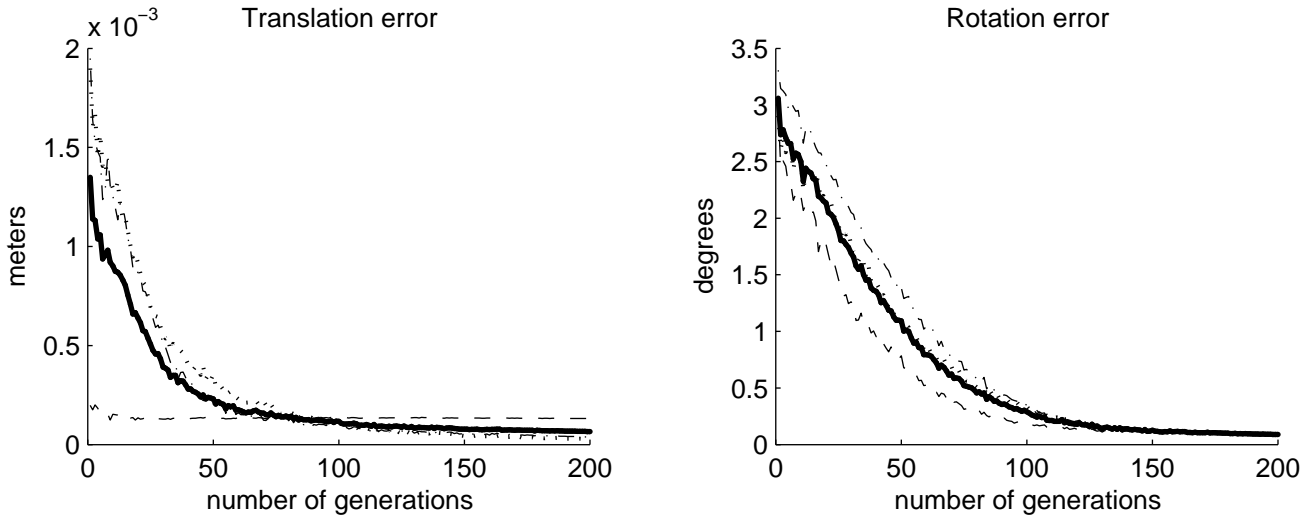


Fig. 6. A figure showing the convergence of the rotation and translation parameters. The plot on the left illustrates the translation errors versus the number of generations. The bolded solid line is the average error of the 3 translation parameters. The dash dotted line, dotted line and the dash line correspond to t_x , t_y and t_z respectively. The plot on the right illustrates the rotation errors versus the number of generations. The bolded solid line is the average error of the 3 rotation angles. The dash dotted line, dotted line and the dash line correspond to the Yaw, Pitch and Roll angle respectively.

can be performed in an individual microprocessor. It means that a maximum of 150 times of speedup can be achieved with 150 processors since the population size is 150. So, it is believed that the proposed approach is applicable to augmented reality.

2) *Pose tracking with our genetic algorithm:* Our genetic algorithm was used to track the pose of an object in a synthetic image sequence. In the sequence, the object was moving with a steady motion at a rate of $[0, 1, 1]$ degrees and $[0.0005, 0.0005, 0.0005]$ meters per frame for [Yaw Pitch Row] and $[t_x, t_y, t_z]$ respectively. Random noise of 0.2 degrees was added to each rotation angle and a noise of 0.0001 meters was added to each translation parameter. The aim of adding random noises is to simulate the non-smooth motion of the movement. The total number of frames in the sequence is 50. For each frame in the sequence, a single search was applied.

Figure 7 shows the results of pose tracking with our algorithm. You can see that the performance is quite good under the conditions that 30 percents of outliers are present. A larger part of the error comes from the Yaw angle, Pitch angle and t_z translation parameter. This kind of ambiguities is due to the fact that these three pose parameters are less observable than the others.

3) *Studying the robustness of the algorithm:* This experiment aims to show the robustness of our algorithm. To do this, we tested our algorithm with the presence of point mismatches and outliers. The results were compared with the genetic algorithm by Hati and Sengupta [2], the traditional Lowe's method described in [23] and a conventional genetic algorithm plus the RANSAC robust estimator. For fairness,

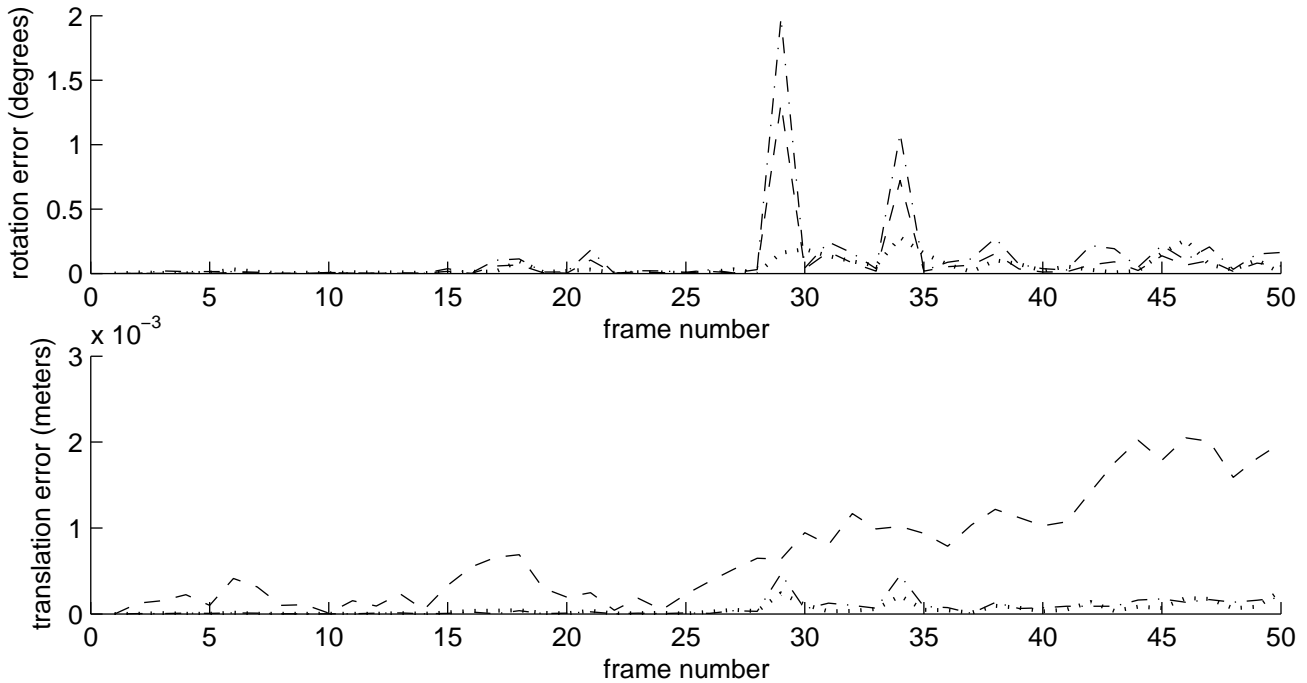


Fig. 7. A figure showing the error of the pose parameters versus frame number in the experiment. The plot on the top shows the rotation errors. The dash line, the dotted line and dash dotted line are the errors of the Roll, Pitch and Yaw angle respectively. The plot at the bottom shows the translation errors. The dash dotted line, dotted line and the dash line are the errors of t_x , t_y and t_z respectively.

these methods were re-implemented using Matlab to ensure that they were compared under the same conditions. The sampling size of the RANSAC estimator is 20 feature points.

The data used in this experiment is synthetic. The specification of the synthetic object and camera settings are described in Section IV A 1) except that special conditions such as point mismatches were added to each test. The test for each condition was repeated 50 times with independent data sets.

a) Effects of point mismatches: In a real situation, it is quite common that some features are mistracked, resulting in wrong point correspondences between the model points and the image points. This experiment is to simulate the problem in such a situation. The point mismatches in the experiment were generated as follows. A number of point pairs were selected randomly from the model features. Their correspondences with the points in the 2D image were swapped. The relationship between the pose errors and the number of point mismatches was studied.

Figure 8 shows the effects of point mismatches to our algorithm. You can see that the pose errors of our algorithm do not depend on the percentage of mismatches. When the feature mismatch percentage is low, say smaller than 20%, the overall performance of our algorithm remains unchanged. The plots in figure 9 show a direct comparison of the four methods under the presence of point mismatches. It is obvious that our algorithm outperformed the other methods. For the rotation error, our algorithm achieves

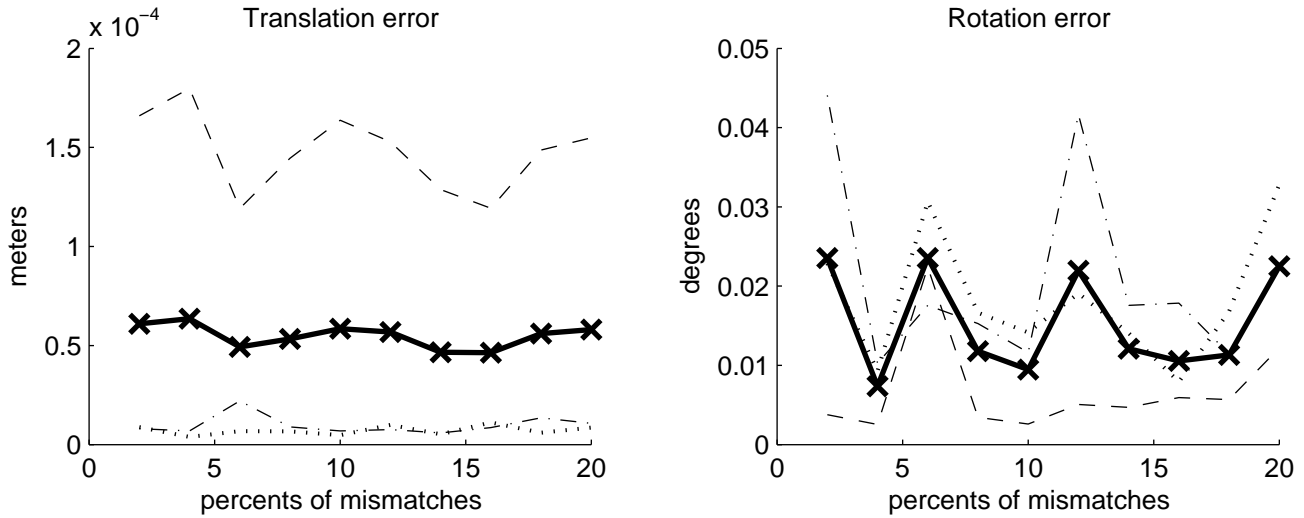


Fig. 8. Graphs showing the effects of point mismatches to our genetic algorithm. The left one shows the translation errors against the number of point mismatches. The solid line with markers 'X' is the average translation error. The dash dotted line, dotted line and the dash line are the errors of t_x , t_y and t_z respectively. The right one shows the rotation errors. The solid line with markers 'X' is the average rotation error of the 3 angles. The dash line, the dotted line and dash dotted line are the errors of the Roll, Pitch and Yaw angle respectively.

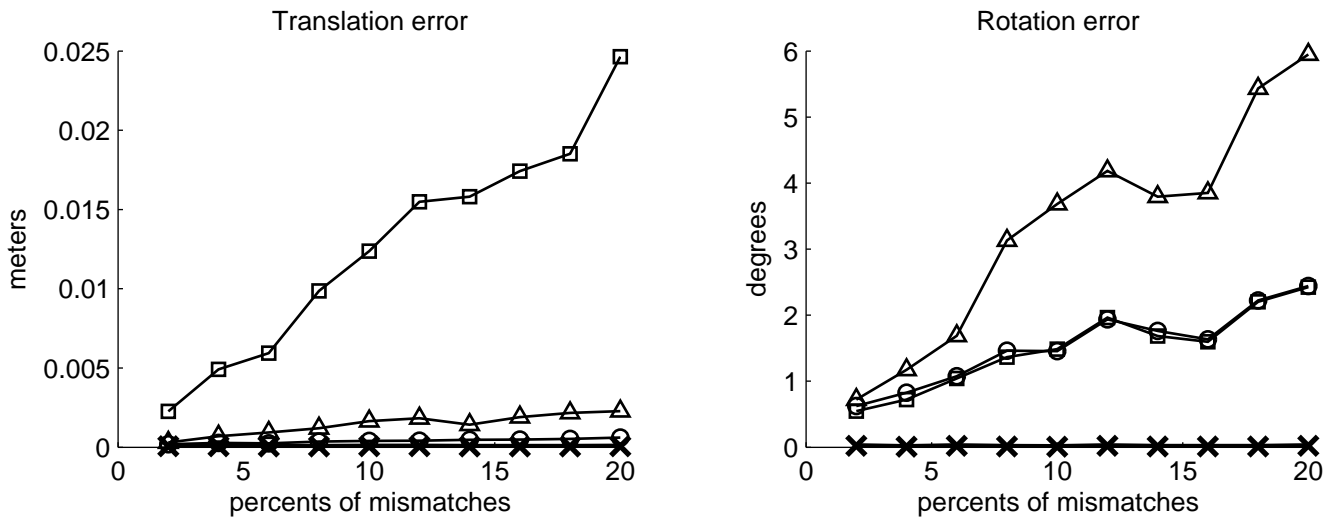


Fig. 9. Graphs showing the comparisons of the accuracy among the four algorithms with the presence of point mismatches. The left plot shows the translation errors while the right one shows the rotation errors. The lines with markers 'X', 'O', '[' and 'Δ' are the results of our approach, the GA by Hati and Sengupta, Lowe's method and a conventional GA plus the RANSAC robust estimator respectively.

an error approximately 0.03 degrees even for 20 percents of mismatches. The other algorithms have error more than 0.5 degrees even for 2 percents of mismatches. For translation error, our algorithm has only 0.05mm error while the others have errors more than 0.19mm. You may also notice that the pose errors increase with the addition of the percentage of wrong correspondences for both the genetic algorithm by Hati and Sengupta [2] and Lowe's method [23].

b) Effects of outliers: Occlusion and disocclusion of a point feature in the image sequence or extracting a point feature from a reflective surface by feature trackers may cause the presence of incorrect

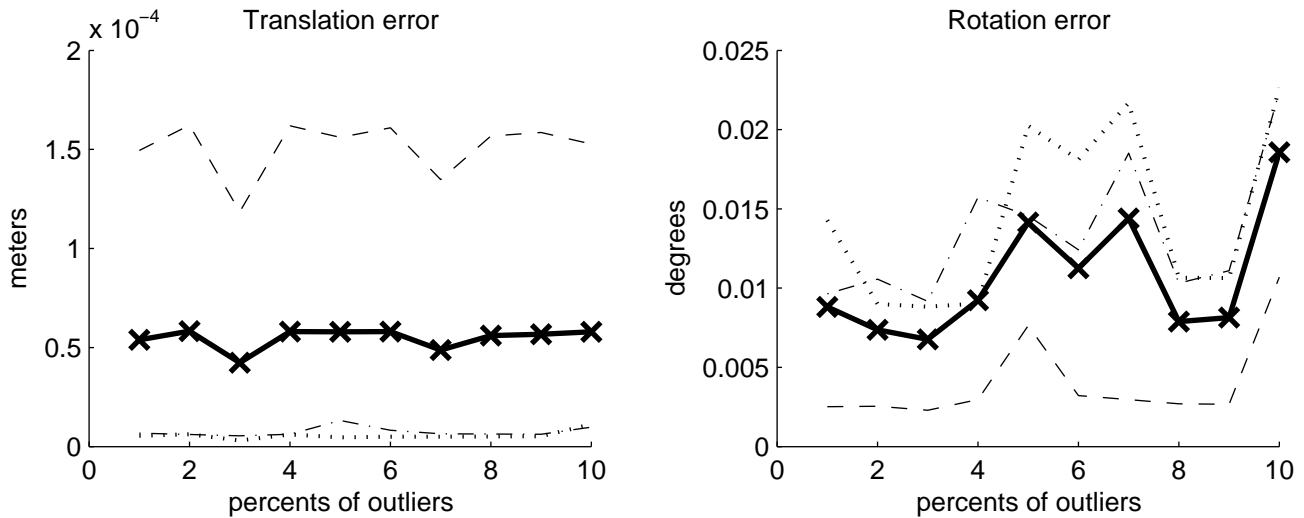


Fig. 10. Graphs showing the effects of outliers to our genetic algorithm. The left one shows the translation errors against the percentage of outliers present. The solid line with markers 'X' is the average translation error. The dash dotted line, dotted line and the dash lines are the errors of t_x , t_y and t_z respectively. The right one shows the rotation errors. The solid line with markers 'X' is the average rotation error of the 3 angles. The dash line, dotted line and dash dotted line and are the errors of the Roll, Pitch and Yaw angle respectively.

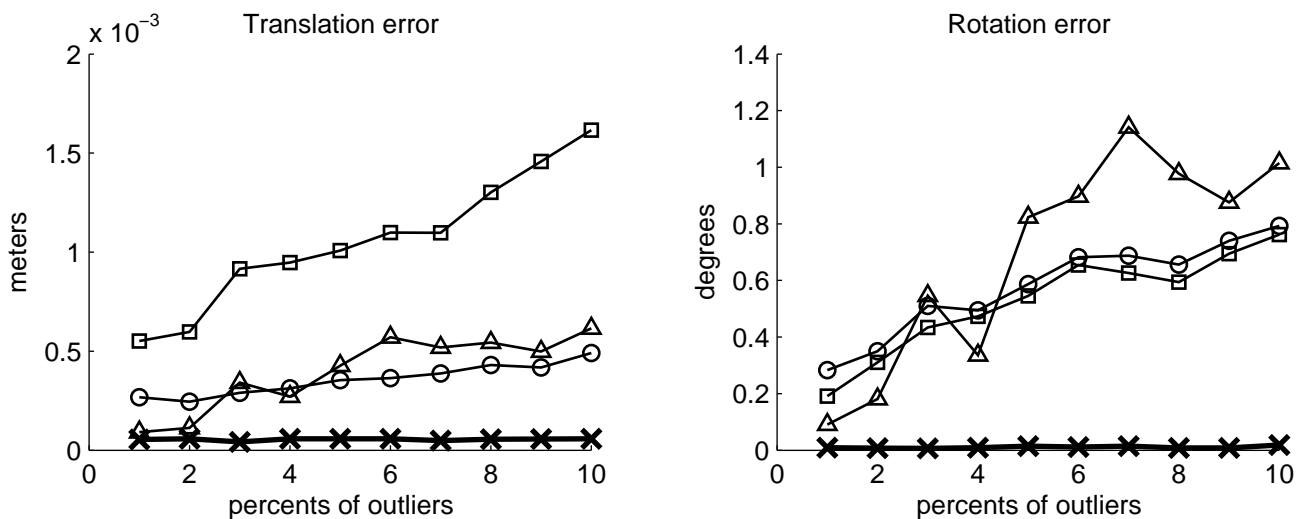


Fig. 11. Graphs showing the comparisons of the accuracy among the three algorithms with the presence of outliers. The left plot shows the translation errors while the right one shows the rotation errors. The lines with markers 'X', 'O', '[]' and '△' are the results of our approach, the GA by Hati and Sengupta, Lowe's method and a conventional GA plus the RANSAC robust estimator respectively.

point features in the images. This results in an inaccurate estimation of the object's pose. The relationship of the pose error and the percentage of outliers present in the images was studied in this experiment. The outliers in the experiment were generated by randomly choosing a number of model features and a high level of random noise was added to their corresponding points in the images. The noise deviates randomly from the range within 100 pixels along the x and y axis on the image plane.

Figure 11 shows the effects of outliers to our algorithm. It is shown that our algorithm has a lower error in estimating both the rotation and translation parameters than the other algorithms. The average

angular error of our algorithm falls below 0.02 degrees for different percentage of outliers but the errors of the other methods are higher than 0.09 degrees even for 1 percent of outliers. The average translation error of our approach is below 0.1mm while the other two algorithms have errors more than 0.1mm. The estimation errors of our algorithm remain approximately the same with the increase in the percentage of outliers. Besides, the incorporation of the RANSAC robust estimator into the conventional genetic algorithm does not have a definite advantage in gaining the accuracy of recovered pose. It is because the algorithm can only make use of a small number of point features in the 3D model.

c) Summary: From the previous experiments, we see that our algorithm outperformed the other algorithms under the presence of point mismatches and outliers. Even the percentages of mismatches or outliers are small (2% out 100 points), the proposed approach also has a lower error than the others. If the point correspondences are all correct, which is very rare in real situations, our algorithm should have a similar performance as the algorithm by Hati and Sengupta, except the computation overhead in processing the second section of the composite chromosome. Since our approach could converge with a population size of 150, which is half of the value of Hati's approach, the computation speed of our algorithm is much higher than that of Hati.

Apart from the algorithm's accuracy, the major disadvantage of the RANSAC-based conventional genetic algorithm is the long computation time. It takes several hundreds random samples from the group of available point features in order to remove the outliers. This means it needs to repeat the genetic search for more than a hundred times. It requires much more computation resources than our algorithm. Among the three genetic algorithm approaches, our algorithm achieves the highest speed. This is probably due to the design of the composite chromosome, which allows the simultaneous search of pose parameters and reliable point features.

B. Experiments with real images

An experiment using real scene images was also performed. Our algorithm has been applied to estimate the pose of an object in a real image sequence. The accuracy of the recovered pose is demonstrated by putting a synthetic chair into the real scene, which is an interesting application of augmented reality.

The procedure of the experiment is as follows. The test image sequence was taken while a grid pattern was on a rotating turntable. This pattern was drawn by an imaging software such that its exact dimension is known. Images were captured using a commercial web camera at a constant time interval. The camera has a fixed focal length and was calibrated using the tool in [22]. The length of the image sequence is

30 frames. The KLT tracker described in [21] was used to extract feature points and track them in the image sequence. With the tracker, features are selected as described by Tomasi and Kanade [20] and points between two frames are matched on a 2D basis. In our experiment, it is assumed that the problem of feature tracking has been solved.

With the known dimension of the pattern and the tracked point features, our genetic algorithm was applied to trace its pose in the image sequence. To place the virtual object into the scene, three feature points on the pattern in the first image were selected to define a plane. With that, the synthetic chair could be placed on this plane using the pose sequence computed by our algorithm.

The results of the experiment are shown in figure 12, 13 and 14. To see the demonstration video, you can refer to the attachment “demo_movie.mpeg” or the URL <http://www.cse.cuhk.edu.hk/~vision/demo/posega/>. Figure 13 shows the pictures that a synthetic chair, which is drawn in a blue wire-frame, was put on the top of the turntable using our genetic algorithm. You can see that in the video the motion of the virtual chair is consistent with the grid pattern. Figure 14 shows the recovered pose parameters. The results are reasonable. You may notice that the lines for translation parameters are relatively flat. This is due to the fact that the grid pattern mainly underwent a rotation motion.

V. FUTURE WORK

One direction for further development is to extend the current algorithm to deal with the structure from motion problem. Both the 3D model and the pose of the object is recovered from 2D images without any a prior knowledge of the object structure. We can extend our genetic algorithm to search for both the structure and the pose of the object. The structure from motion problem suffers from shape and pose ambiguities [7] with many of the greatest descent searches. Genetic algorithm is useful to deal with this problem as it can avoid locking into a local optimum.

VI. CONCLUSION

A new method for computing the pose of an object based on the genetic algorithm framework is proposed in this paper. In our method, we search for the set containing the most reliable feature points of an object in addition to the its pose in the image sequence. Our approach has the advantage of avoiding the local optimal solutions compared to the traditional greatest descent methods. Our algorithm is also efficient in excluding outliers and point mismatches.

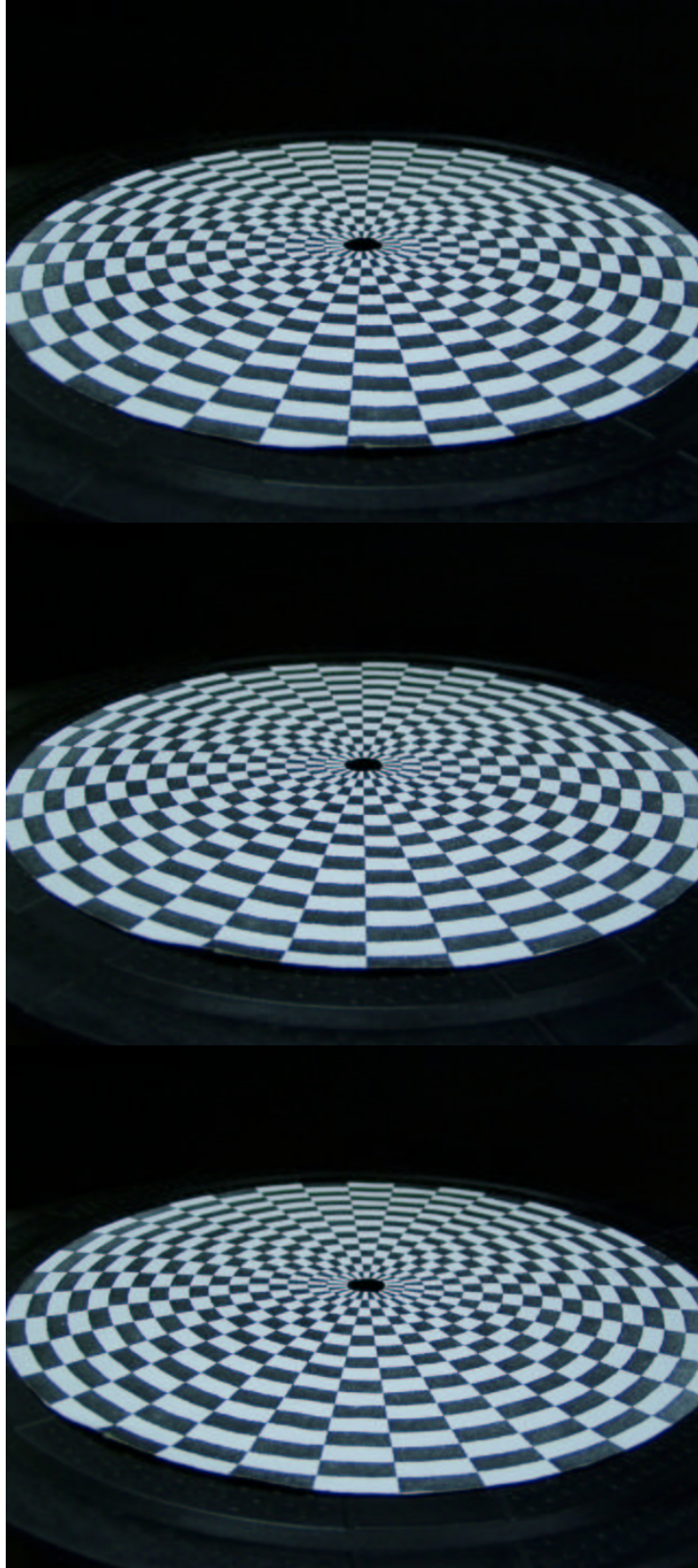


Fig. 12. The 1st, 15th and 30th image (from top to down) in the original image sequence in the real scene experiment.

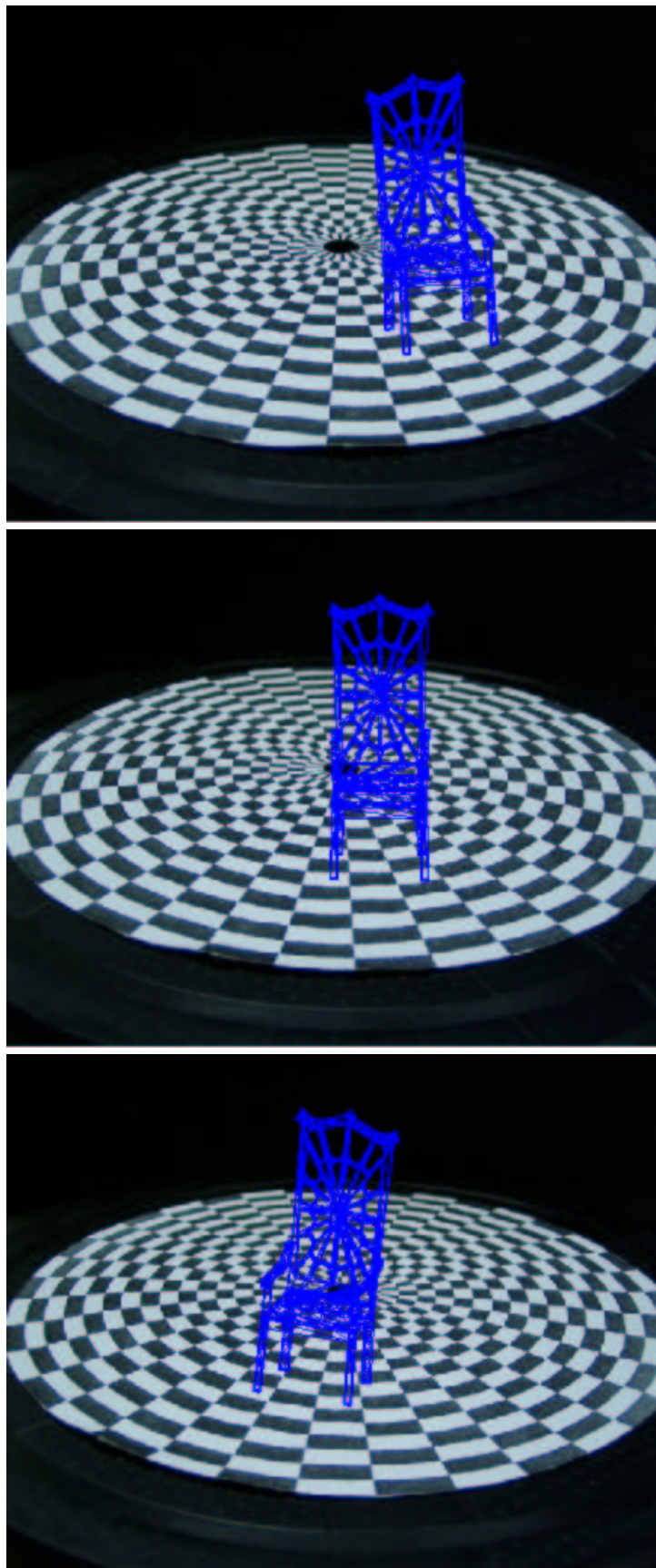


Fig. 13. The results of putting the synthetic chair into the image sequence in the real image experiment. The top, the middle and the bottom picture are the 1st, 15th and 30th image in the processed sequence respectively. To see the video in the above demonstration, please refer to the attachment “demo_movie.mpeg”. It can be also be found at <http://www.cse.cuhk.edu.hk/~vision/demo/posega/>

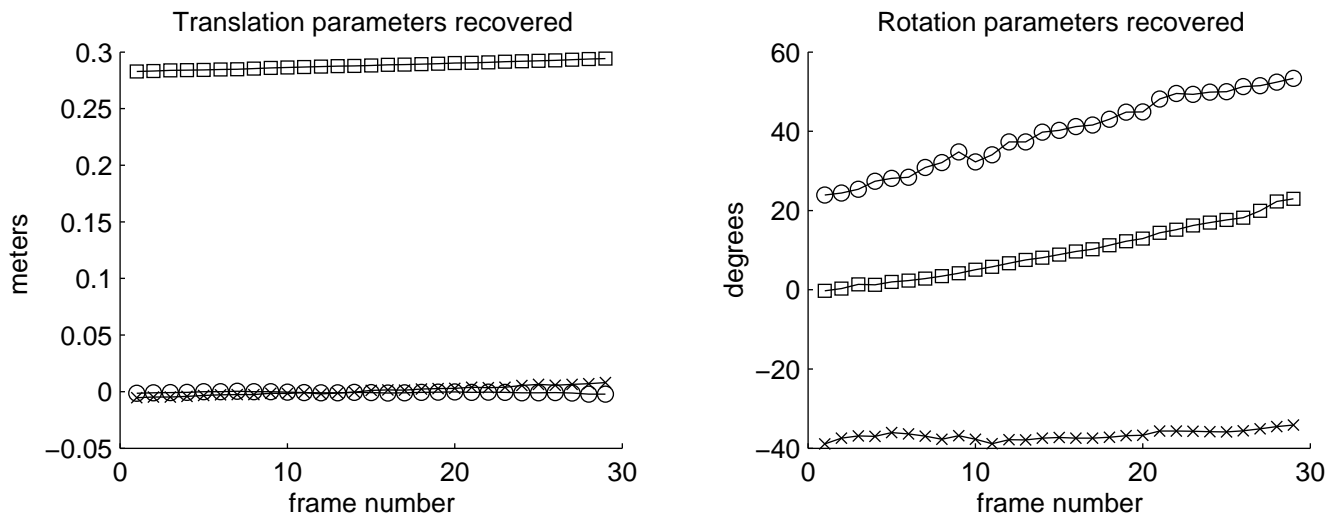


Fig. 14. Graphs showing the pose parameters recovered in the real scene experiment. The plot on the left shows the translation parameters recovered in meters. The lines with markers 'X', 'O' and '[]' correspond to t_x , t_y and t_z respectively. The plot on the right shows the rotation parameters recovered in degrees. The lines with markers 'X', 'O' and '[]' correspond to the Yaw, Pitch and Roll angle respectively.

Experimental results show that there is a significant improvement on the pose estimation accuracy with our feature searching strategy. Our genetic algorithm outperformed three other approaches under the presence of point mismatches and outliers. Real image experiment has been performed and our algorithm has been applied to create an augmented reality movie. The result is good in general. The motion of both the real scene and the virtual object is consistent in our demonstration video.

To pursue further, we would like to extend our algorithm to tackle the structure from motion problem. Moreover, the idea of feature searching in our algorithm is not merely limited to vision-based pose estimation. It can be extended to solve the camera calibration problem or marker-based pose estimation for augmented reality to reduce the problem caused by noise.

ACKNOWLEDGMENT

The work described in this paper was supported by a grant from the Research Grant Council of Hong Kong Special Administrative Region. (Project Number. CUHK 4204/04E)

REFERENCES

- [1] Goldberg, D.D., "Genetic algorithm in search, optimization and machine learning", Addison-Wesley, Reading, MA, Wokingham, 1989.
- [2] S.Hati, S.Sengupta, "Robust camera parameter estimation using genetic algorithm", Pattern Recognition Letters 22, page 289-298, 2001.
- [3] Fubito Toyama, Kenji Shoji and Juichi Miyamochi, "Model-based pose estimation using genetic algorithm", International Conference on Pattern Recognition 98, page 198-201, 1998.
- [4] Qiang Ji and Yongmian Zhang, "Camera calibration with genetic algorithm", IEEE Transaction on Systems, Man and Cybernetics-Part A:Systems and Humans, Vol. 31, No. 2, March 2001.

- [5] P.Cerveri, A.Pedotti and N.A Borghese, "Combined evolution strategies for dynamic calibration of video-based measurement systems", IEEE Transaction on Evolutionary Computation, Vol. 5, No. 3, June 2001.
- [6] Emanuele Trucco, Alessandro Verri, "Introductory Techniques for 3-D Computer Vision", Prentice Hall, 1998.
- [7] R.Szeliski and S.B.Kang, "Shape ambiguities in structure from motion", IEEE Transaction on Pattern Analysis and Machine Intelligence, vol 19, no.5, May 1997.
- [8] K.H. Wong, S.H. Or and M.M.Y. Chang, "Pose tracking for virtual walk-through environment construction", Proceedings of the Recent Development in Theories and Numerics, International Conference on Inverse Problems, Hong Kong, China, January 9-12, 2002, pp. 394 - 402, eds: Y.C. Hon, M. Yamamoto, J. Cheng and J.Y. Lee, World Scientific Publishing Co. Pte. Ltd. 2003.
- [9] Mohinder S.Grewal, Angus P.Andrews, "Kalman Filtering Theory and Practice", Prentice Hall, 1993.
- [10] S.Tsutsui and Y.Fujimoto. "The-p-fGA: Phenotypic forking genetic algorithm", JSAI, 11 (4), page 619-628, 1996.
- [11] P.A. Beardsley, A.Zisserman and D.W.Murray, "Sequential updating of projecting and affine structure from motion", International Journal of Computer Vision 23, pp235-259, 1997.
- [12] Vincenzo Lippiello, Bruno Siciliano and Luigi Villani, "Objects motion estimation via BSP tree modeling and Kalman filtering of stereo images", Proceedings of the IEEE International Conference on Robotics and Automation Washington DC, pages 2968-2973, 2002.
- [13] Vincenzo Lippiello, Bruno Siciliano and Luigi Villani, "Position and orientation estimation based on Kalman filtering of stereo images", Proceedings of the IEEE International Conference on Control Applications Mexico City, pages 702-707, 2001.
- [14] Chang M.Y.Y. and Wong K.H., "Model reconstruction and pose acquisition using extended Lowe's method", IEEE Transactions on Multimedia (accepted for publication, Sept. 2003).
- [15] Xiang Zhang, Stephan Fronz and Nassir Navab, "Visual marker detection and decoding in AR systems: A comparative study", Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR'02), 2002.
- [16] Jun Park, Bolan Jiang and Ulrich Neumann, "Vision-based pose computation: robust and accurate augmented reality tracking", IEEE International Workshop on Augmented Reality, pp 3-12, Oct. 1999.
- [17] B.Triggs, P.McLauchlan, R.Hartley and A.Fitzgibbon, "Bundle adjustment V A modern synthesis" In proceedings of the International Workshop on Visual Algorithm: Theory and Practice. Pp 298-372, Corfu Greece, 1999.
- [18] Martin A. Fischler and Robert C.Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography", Communications of the ACM, Vol24, pp.882-887, no.6, June 1981.
- [19] Radu Horaud, Bernard Conio and Oliver Le Boulleux, "An analytic solution for the perspective 4-point problem", Computer Vision, Graphics and Image Processing 47, pages 33-44, 1989.
- [20] Carlo Tomasi and Takeo Kanade, "Detection and tracking of point features", Carnegie Mellon University Technical Report CMU-CS-91-132, April 1991.
- [21] KLT: An Implementation of the Kanade-Lucas-Tomasi Feature Tracker (<http://vision.stanford.edu/birch/klt/>).
- [22] Camera Calibration Toolbox for Matlab (<http://www.vision.caltech.edu/bouguetj/calib.doc/>).
- [23] David G. Lowe, "Fitting parameterized three-dimensional models to images", IEEE Pattern Analysis and Machine Intelligence, Volume: 13 Issue: 5, Page(s): 441 -450, May 1991.
- [24] Simon Gibson, Jon Cook, Toby Howard, Roger Hubbold and Dan Oram, "Accurate camera calibration for off-line, video-based augmented reality", IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR 2002), Darmstadt, Germany, September 2002.
- [25] X. Zhang, Y. Genc, and N. Navab, "Taking AR into large scale industrial environments: Navigation and information access with mobile computers" In IEEE Int.Symp.on Augmented Reality, 2001.

- [26] Liu M.L. and Wong K.H., "Pose estimation using four corresponding points", Pattern Recognition Letters, Volume 20, Number 1, pp.69-74, January 1999.
- [27] Or S.H., Luk W.S., Wong K.H. and King I., "An efficient iterative pose estimation algorithm", Image and Vision Computing Journal, Volume 16, Issue 5, pp.355-364, May 1998.
- [28] Marc Pollefeys, "Tutorial on 3D Modeling from Images", In conjunction with ECCV, June 2000.

Mr. Ying Kin Yu received a B.Eng (first class honours) and an M.Phil. degree from the Chinese University of Hong Kong in 2002 and 2004. He is now a PhD student in the Department of Computer Science and Engineering in the same university. He has been awarded the prestigious Sir Edward Youde Memorial Fellowship twice for his academic achievements in his graduate study. His research interests are computer vision, augmented reality, Kalman filtering and genetic algorithms. His contact address is: The Computer Science and Engineering Dept., The Chinese University of Hong Kong, Shatin, Hong Kong . Email: ykyu@cse.cuhk.edu.hk

Prof. Kin Hong Wong received a B.Sc. in Electronics and Computer Engineering from the University of Birmingham in 1982, and a Ph.D. from the Engineering Dept. of the University of Cambridge, U.K. in 1986. He was a Croucher research fellow at the University of Cambridge from 1985 to 1986. Prof. Wong joined the Computer Science Dept. of CUHK in 1988 and is now an Associate Professor. His research interests are 3D computer vision, virtual reality image processing, pattern recognition, microcomputer applications and computer music. His contact address is: The Computer Science and Engineering Dept., The Chinese University of Hong Kong, Shatin, Hong Kong . Email: khwong@cse.cuhk.edu.hk

Prof. Michael Ming Yuen Chang received the B.Sc. in electrical engineering from Imperial College, London University and the PhD degree in electrical engineering from University of Cambridge in 1988. He then joined the Department of Information Engineering, The Chinese University of Hong Kong and is now an Associate Professor. His current research interest is in character recognition, scientific visualization and intelligent instrumental control. His contact address is: The Information Engineering Dept., The Chinese University of Hong Kong, Shatin, Hong Kong . Email: mchang@ie.cuhk.edu.hk