Camera Calibration with Genetic Algorithms

Qiang Ji and Yongmian Zhang

***Abstract*—In this paper, we present a novel approach based on genetic algorithms for performing camera calibration. Contrary to the classical nonlinear photogrammetric approach [1], the proposed technique can correctly find the near-optimal solution without the need of initial guesses (with only very loose parameter bounds) and with a minimum number of control points (7 points). Results from our extensive study using both synthetic and real image data as well as performance comparison with Tsai's procedure [2] demonstrate the excellent performance of the proposed technique in terms of convergence, accuracy, and robustness.**

***Index Terms*—Camera calibration, computer vision, genetic algorithms, nonlinear optimization.**

I. Introduction

CAMERA calibration is an essential step in many machine vision and photogrammetric applications including robotics, three-dimensional (3-D) reconstruction, and mensuration. It addresses the issue of determining intrinsic and extrinsic camera parameters using two–dimensional (2-D) image points and the corresponding known 3-D object points (hereafter referred to as control points). The computed camera parameters can then relate the location of pixels in the image to object points in the 3-D reference coordinate system. The existing camera calibration techniques can be broadly classified into linear approaches [3]–[8] and nonlinear approaches [3]–[5], [6], [9], [10]. Linear methods have the advantage of computational efficiency but suffer from a lack of accuracy and robustness. Nonlinear methods, on the other hand, offer a more accurate and robust solution but are computationally intensive and require good initial estimates. For example, the classical nonlinear photogrammetric approach [1] will not correctly converge if the initial estimates are not very close. Hence, the quality of an initial estimate is critical for the existing nonlinear approaches. In most photogrammetry situations, auxiliary equipment can often provide approximate estimates of camera parameters. For example, scale and distances are often known to within 10% and angle is known to be within 15° [11].

For computer vision problems, however, approximate solutions are usually not known *a-priori*. To get around this problem, one common strategy in computer vision is to attack the camera calibration problem by using two steps [2], [12]. The first step generates an approximate solution using a linear technique, while the second step improves the linear solution using a nonlinear iterative procedure.

Manuscript received April 13, 1999; revised January 5, 2001. This paper was recommended by Associate Editor W. Pedrycz.

Q. Ji is with the Department of Electrical, Computer, and System Engineering, Rensselaer Polytechnic Institute, Troy, NY 12180 USA (e-mail: qji@ecse.rpi.edu).

Y. Zhang is with the Department of Computer Science, University of Nevada, Reno, NV 89557 USA.

Publisher Item Identifier S 1083-4427(01)02335-9.

The first step utilizing linear approaches is key to the success of two-step methods. Approximate solutions provided by the linear techniques must be good enough for the subsequent nonlinear techniques to correctly converge. Being susceptible to noise in image coordinates, the existing linear techniques are, however, notorious for their lack of robustness and accuracy [13]. Haralick *et al.* [14] shows that when the noise level exceeds a knee level or the number of points is below a knee level, these methods become extremely unstable and the errors skyrocket. The use of more points can help alleviate this problem. However, fabrication of more control points often proves to be difficult, expensive, and time-consuming. For applications with a limited number of control points, e.g., close to the required minimum number, it is questionable whether linear methods can consistently and robustly provide good enough initial guesses for the subsequent nonlinear procedure to correctly converge to the optimal solution.

Another problem is that almost all nonlinear techniques employed in the second step use variants of conventional optimization techniques like gradient-descent, conjugate gradient descent or the Newton method. They therefore all inherit well known problems plaguing these conventional optimization methods, namely, poor convergence and susceptibility to getting trapped in local extrema. If the starting point of the algorithm is not well chosen, the solution can diverge, or get trapped at a local minimum. This is especially true if the objective function landscape contains isolated valleys or broken ergodicity. The objective function for camera calibration involves 11 camera parameters and leads to a complex error surface with the desired global minimum hidden among numerous finite local extrema. Consequently, the risk of local rather than global optimization can be severe with conventional methods.

To alleviate the problems with the existing camera calibration techniques, we explore an alternative paradigm based on genetic algorithms (GAs) to conventional nonlinear optimization methods. GAs were designed to efficiently search a large, nonlinear, poorly understood spaces and have been widely applied in solving difficult search and optimization problems including camera calibration [15], spectrometer calibration [16], instrument and model calibration [17], [18]. GAs have attractive features for camera calibration problems because they do not require specific models or linearity as in classical approaches and can explore all parts of the feasible uncertainty-parameter space. Compared with the conventional nonlinear optimization techniques, the GAs offer the following key advantages:

1) *Autonomy*: GA does not require an initial guess. The initial parameter set is generated randomly in the predefined parameter domain.
2) *Robustness*: Conceptually, GA works with a rich population and simultaneously climbs many peaks in parallel

during the search process. This significantly reduces the probability of getting trapped at a local minimum.

3) *Noise Immunity*: GA searches a fit parameter set and moves toward the global optimum by gradually reducing the chance of reproducing unfit parameter sets. It therefore has high accuracy potential in noise situation.

Results from our study are encouraging and promising. The proposed GA approach can quickly converge to the correct solution without initial guesses and with the minimum number of points (seven points). We believe that this study is significant for computer vision and photogrammetry in that

- the proposed technique does not require good initial guesses of camera parameters.
- the proposed technique is robust and accurate even with the minimum number of control points for noisy images.

The remainder of this paper is organized as follows. In Section II, we provide a brief introduction to the basic operations of the genetic algorithms and to the perspective geometry for camera calibration. Section III defines the fitness function for the camera calibration problem and Section IV describes how genetic algorithms can be adapted to camera calibration problems. Section V discusses the convergence and computational complexity of this technique. We present experimental results and analysis in Section VI. Conclusions and summary are presented in Section VII.

II. Background

To offer the necessary background, in this section we provide short introductions to genetic algorithms and the perspective geometry used for camera calibration.

A. Genetic Algorithms

GAs are stochastic, parallel search algorithms based on the mechanics of natural selection and the process of evolution [19]. GAs were designed to efficiently search large, nonlinear spaces where expert knowledge is lacking or difficult to encode, and where traditional optimization technique fail. GAs perform a multidirectional search by maintaining a population of potential solutions and encourage information formation and exchange between these solutions. A population is modified by the probabilistic application of the genetic operators from one generation to the next.

The three basic genetic operations in GAs are 1) evaluation; 2) selection; and 3) recombination, as shown in Fig. 1. Evaluation of each string which encodes a candidate solution is based on a fitness function. This corresponds to the environmental determination of survivability in natural selection. Selection is done on the basis of relative fitness and it probabilistically culls solutions from the population that have relatively low fitness. Two candidate solutions (p_i and p_j) with high fitness are then chosen for further reproduction.

Selection serves to focus search into areas of high fitness. Of course, if selection were the only genetic operator, the population would never have any individuals other than those introduced in the initial population. New population is generated by perturbing the current solutions via *recombination*. Recombination, which consists of mutation and crossover,

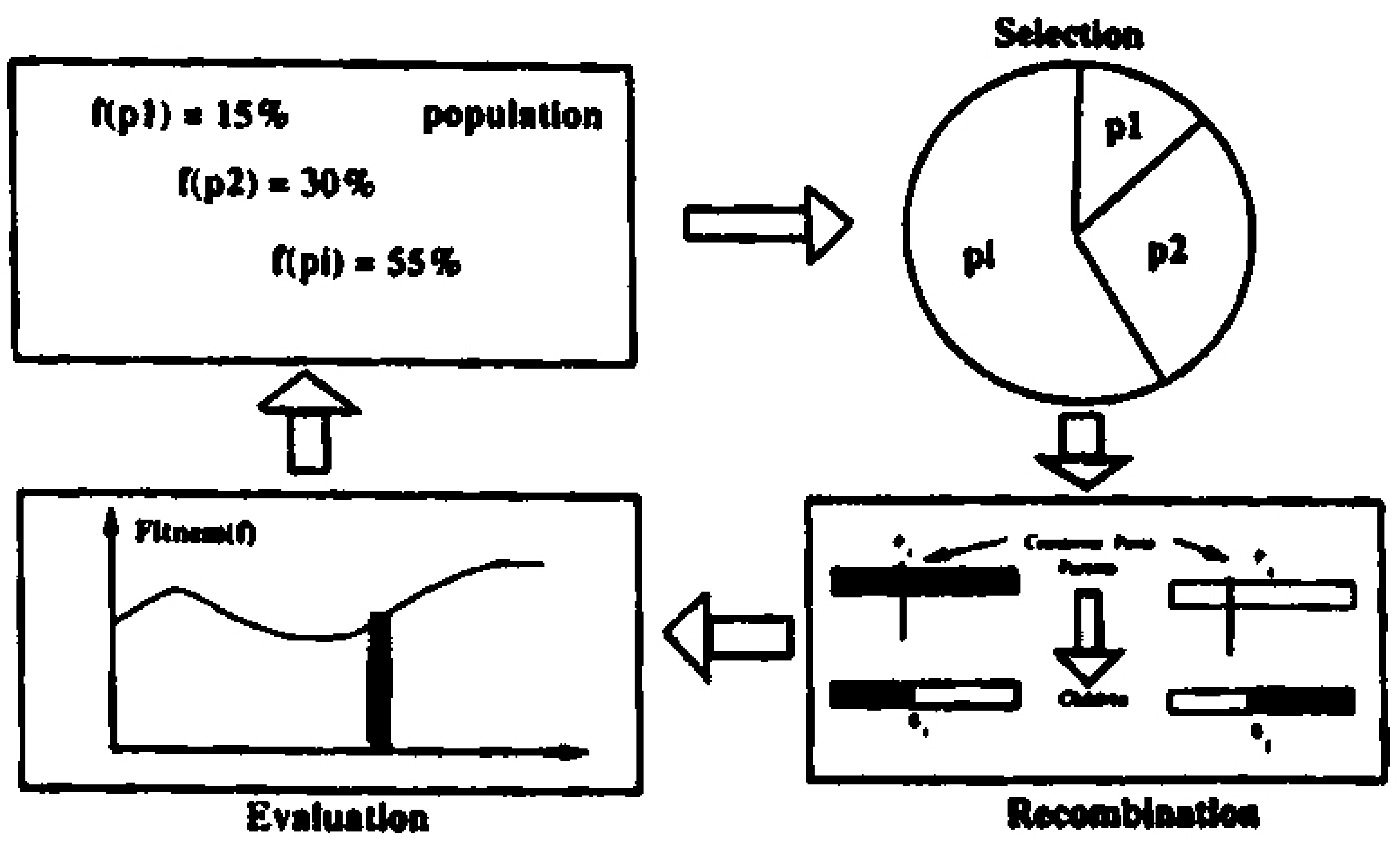

Fig. 1. Three basic genetic operations.

imitates sexual reproduction. Crossover is a structured yet stochastic operator that allows information exchange between candidate solutions. The simplest way to perform crossover is to choose randomly some crossover point and everything before this point copy from a first parent and then everything after a crossover point copy from the second parent as shown in Fig. 1.

Mathematically, crossover follows that

$$c_{new} = \alpha p_i + \beta p_j \tag{1}$$

where

p_i and p_j	parent individuals from the last iteration/generation;
c_{new}	new individual in the current generation;
α and β	the proportion of good alleles[1] which may be probabilistically inherited from p_i and p_j.

After a crossover is performed, mutation takes place. This is to prevent falling all solutions in population into a local optimum of solved problem. The mutation operator introduces new genetic structures in the population by randomly changing some of its building blocks, helping the algorithm escape local minima traps. It is clear that the crossover and mutation are the most important part of the genetic algorithm. The performance is influenced mainly by these two operators. More introduction on GA and a detailed discussion on mutation and crossover may be found in [20] and [21].

In summary, the operation of the basic GA can be outlined as follows.

1) Generate random population of n chromosomes[2] (suitable solutions for the problem).
2) Evaluate the fitness of each chromosome x in the population.
3) Select two parent chromosomes from a population according to their fitness.
4) With a crossover probability cross over the parents to form a new offspring.
5) With a mutation probability mutate new offspring.
6) Place new offspring in a new population.

[1]An alternative form of a gene that can exist at a single gene position.
[2]A linear end-to-end arrangement of genes.

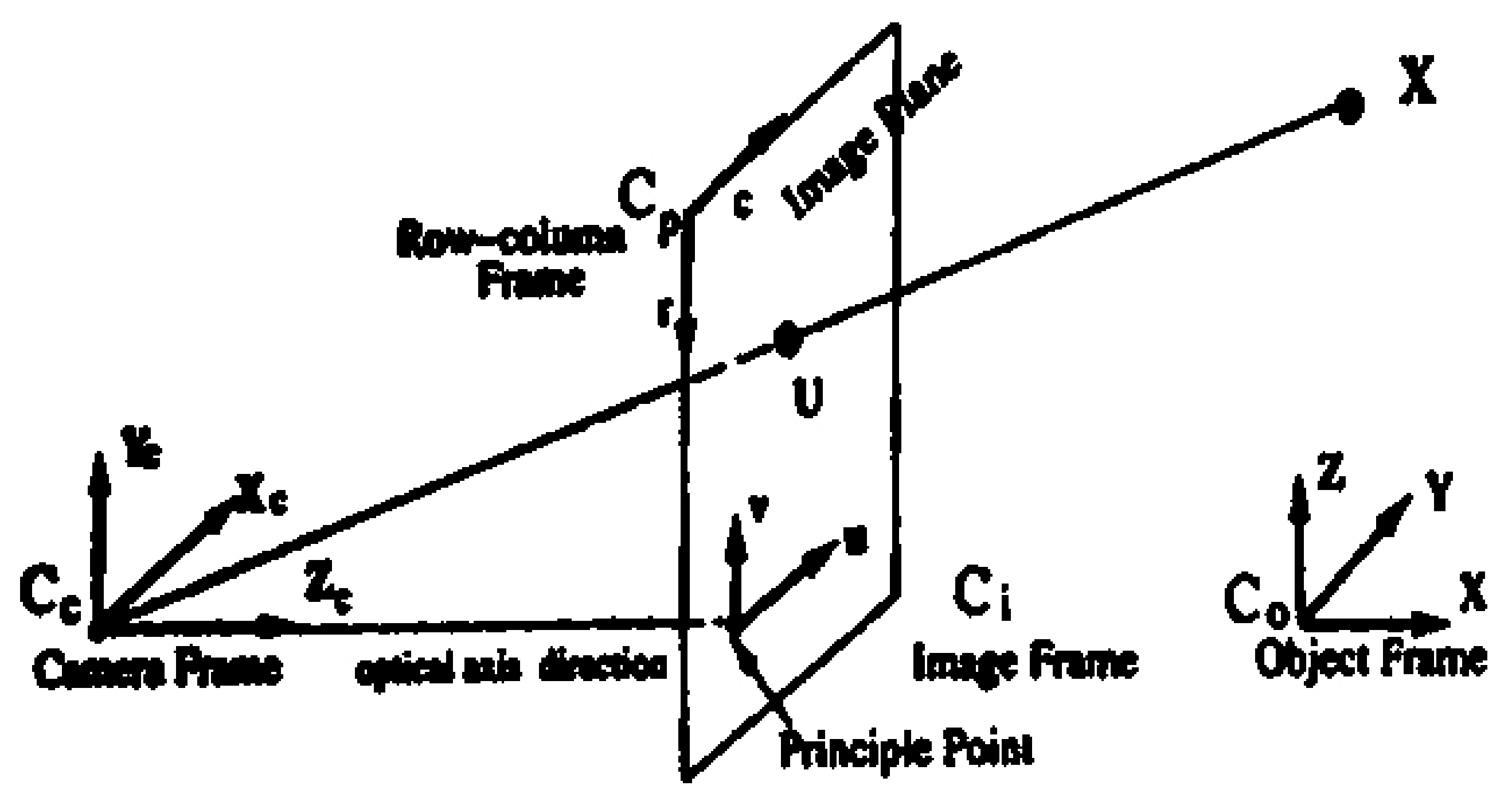

Fig. 2. Perspective projection geometry.

7) Use new generated population for a further run of algorithm (loop from step 2 until the end condition is satisfied.

B. Perspective Geometry

Fig. 2 shows a pinhole camera model and the associated coordinate frames. Let $X = (x\ y\ z)^T$ be a 3-D point in an object frame and $U = (u\ v)$ the corresponding image point in the image frame. Let $X_c = (p\ q\ s)^T$ be the coordinates of X in the camera frame and $p = (c\ r)^T$ be the coordinates of U in the row–column frame as illustrated in Fig. 2. The image plane, which corresponds to the image sensing array, is assumed to be parallel to the (X_c, Y_c) plane of the camera frame and at a distance f to its origin, where f denotes the focal length of the camera. The relationship between the camera frame C_c and object frame C_o is given by

$$X = RX_c + T \tag{2}$$

where R is a 3×3 rotation matrix defining the camera orientation and T is a translation vector representing the camera position. R and T can further be parameterized as

$$R = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \quad T = \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix}. \tag{3}$$

The r_{ij} in matrix R can be expressed as the function of camera pan angle ω, tilt angle ϕ, and swing angle κ as follows:

$$\begin{cases} r_{11} = \cos\phi\cos\kappa \\ r_{12} = \sin\omega\sin\phi\cos\kappa + \cos\omega\sin\kappa \\ r_{13} = -\cos\omega\sin\phi\cos\kappa + \sin\omega\sin\kappa \\ r_{21} = -\cos\phi\sin\kappa \\ r_{22} = -\sin\omega\sin\phi\sin\kappa + \cos\omega\cos\kappa \\ r_{23} = \cos\omega\sin\phi\sin\kappa + \sin\omega\cos\kappa \\ r_{31} = \sin\phi \\ r_{32} = -\sin\omega\cos\phi \\ r_{33} = \cos\omega\cos\phi. \end{cases} \tag{4}$$

The collinearity of 3-D object coordinate X and 2-D image coordinate p can be written as

$$\begin{cases} c = fs_x \dfrac{r_{11}x + r_{12}y + r_{13}z + t_x}{r_{31}x + r_{32}y + r_{33}z + t_z} + u_0 = g(X, \mathrm{q}) \\ r = fs_y \dfrac{r_{21}x + r_{22}y + r_{23}z + t_y}{r_{31}x + r_{32}y + r_{33}z + t_z} + v_0 = w(X, \mathrm{q}) \end{cases} \tag{5}$$

where

s_x and s_y	scale factors (pixels/mm) due to spatial quantization;
u_0 and v_0	coordinates of the principle point in pixels relative to image frame;
q	vector of all camera parameters as defined in (6).

The main task of camera calibration in 3-D machine vision is to obtain an optimal set of the interior camera parameters $((u_0, v_0), s_x, s_y, f)^T$ and exterior camera parameters $(\omega, \phi, \kappa, t_x, t_y, t_z)^T$ using known control points in the 2-D image and their corresponding 3-D points in the object coordinate system.

III. Objective Function

Let q be an vector consisting of the unknown interior and exterior camera parameters, that is,

$$\mathrm{q} = [u_0, v_0, f, s_x, s_y, \omega, \phi, \kappa, t_x, t_y, t_z]^T. \tag{6}$$

For notational convenience, we rewrite q as $\mathrm{q} = (q_1, q_2, \ldots, q_{11})^T$, where q_1, q_2 and q_{11} correspond to u_0, v_0, and t_z, respectively, in the previous notation used for q. Assume q is a solution of interior and exterior camera parameters and $\mathrm{q} \subseteq \mathrm{Q}$, then we have

$$\mathrm{Q} = \left\{\mathrm{q}\colon q_i \in \left[q_i^-; q_i^+\right];\ i = 1, 2, \ldots, n\right\} \tag{7}$$

where q_i^- and q_i^+ are the lower and upper bounds of q_i. The bounds on parameters can be obtained based on the knowledge of camera. Any reasonable interval which may cover possible parameter values may be chosen as the bound of parameter q_i. For example, we may have $\omega, \phi, \kappa \in [-\pi, \pi]$, $f \in [20, 70]$ as in our test cases and so on. An optimal solution of q with M control points can be achieved by minimizing

$$\sum_{i=1}^{M} \left[(g(\mathrm{q}, X_i) - c_i)^2 + (w(\mathrm{q}, X_i) - r_i)^2\right] \qquad \mathrm{q} \in \mathrm{Q} \tag{8}$$

where $X_i = (x_i, y_i, z_i)$ is the ith 3-D point; g and w are defined in (5).

A key issue that arises in this approach is the extremely large search space caused by the presence of uncertainties. Fig. 3 plots the landscape of the objective function as a function of the three rotation angles with other camera parameters such as $[u_0, v_0, f, s_x, s_y, t_x, t_y, t_z]^T$ fixed. This figure reveals several local extrema with the objective function. It is reasonable to conjecture that, when more parameters need to be calibrated, the number of local extrema increases and the landscape of the objective function becomes more complex. This presents a serious challenge to conventional minimization procedures since there may be several local minima as shown in Fig. 3, and the choice of starting point will determine which minimum the procedure converges to, or whether it will converge at all. If starting points are far away from the desired minimum, the traditional optimization techniques could converge erroneously.

IV. GA Operators and Representation

Designing an appropriate encoding and/or recombination method is often crucial to the success of a GA algorithm. To im-

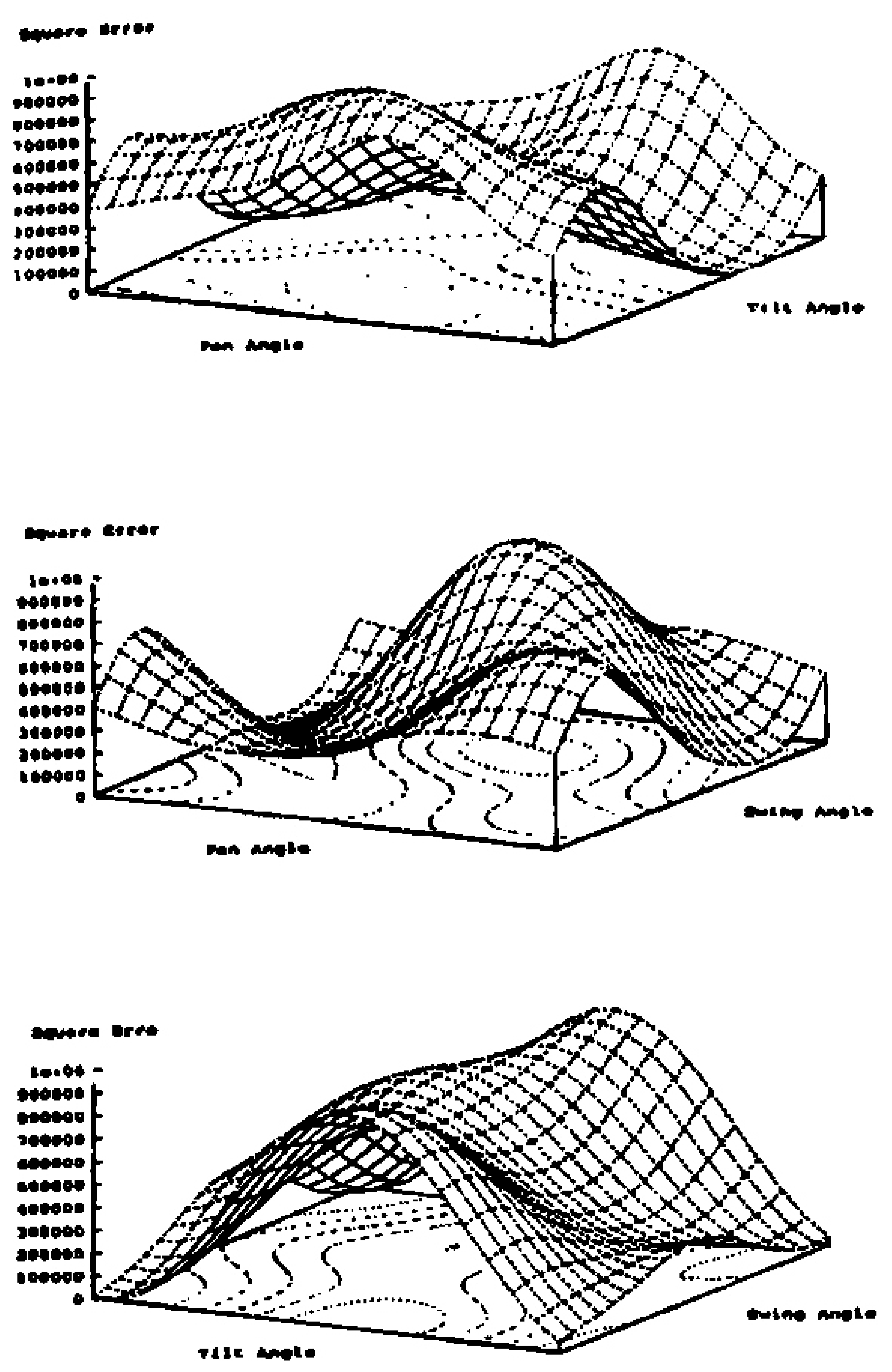

Fig. 3. Landscape of objective function under varying rotation angles assuming other interior and exterior parameters are fixed and $\omega, o, \kappa \in [-\pi, \pi]$.

prove GA's convergence, we propose a *new* mutation operator that determines the amount and direction of perturbation in the search space. Mutation can be viewed as one dimensional (1-D) or local search, while crossover performs multidimension, or more global, search.

A. Representation

GA chromosome are usually encoded as bit strings and a long binary string is required in order to represent a large continuous range for each parameter. Instead, we encode the GA chromosome as a vector of real numbers. Each camera parameter $q_i, i = 1, \ldots, n$ is initialized to a value within its respective bounds as defined in (7). The chromosome vector may be defined as

$$\begin{cases} \mathbf{q}_i^t = (q_1, \ldots, q_k, \ldots, q_n), & 1 \le k \le N \\ \mathbf{q}_i^{t+1} = (q_1, \ldots, q_k', \ldots, q_n), & 1 \le k \le N \end{cases} \tag{9}$$

where

$\mathbf{q}_i^t$ individual from population N at tth generation;

$\mathbf{q}_i^{t+1}$ individual from the new generation after genetic selection;

q_k' parameter that has been modified during the evolutionary process.

B. Mutation

There two types of mutation operators are 1) per-chromosome mutation operator and 2) per-gene mutation operators. Per-chromosome operator acts upon an entire chromosome. Per-gene mutation operator, on the other hand, acts on each gene individually. Our mutation technique belongs to the former. Specifically, we implement a local gradient-descent search technique to identify a new solution nearby but with a higher fitness. Our mutation technique doesn't just tweak individual genes; it alters the chromosome as a whole.

Our mutation scheme comprises two steps: 1) determining the search direction and 2) simultaneously determining the step size in the selected search direction. In (7), the search space $\mathbf{Q}$ should be a convex space S. The task of the GA is to determine an unknown optimal point $q_k^* \in [q_k^-, q_k^+]$ in the convex space S which minimizes the global error function of (8) at that point.

Assuming that the probability of receiving a correct step size from the GA is p, whenever the current $q_k \le q_k^*$ the GA correctly increases q_k with a probability p. It may also, however, incorrectly decrease q_k with a probability $1-p$. It is reasonable to expect that it is equally likely for a GA to increase q_k as to decrease q_k. Then the next current point q_k^{t+1} is given by

$$q_k^{t+1} = \begin{cases} q_k^t + I\Delta(t, q_k^+) + (I-1)\Delta(t, q_k^-) & \\ q_k^+, & \text{if } q_k^{t+1} > q_k^+ \\ q_k^-, & \text{if } q_k^{t+1} < q_k^- \end{cases} \tag{10}$$

where $\Delta(t, q_k^+)$ and $\Delta(t, q_k^-)$ are the step sizes for the upper and lower bounds of q_k, respectively. I is an indicator function assuming the value of 0 and 1 depending on the outcome of a coin toss.

The crucial issue is the amount of perturbation (step size) of point q_k in the interval $[q_k^-, q_k^+]$. Too small perturbation may lead to sluggish convergence, while too large perturbation may cause the GA to erroneously converge or even oscillate. Since $q_k \in [q_k^-, q_k^+] \in S$ holds, q_k must be a fraction, say c, of the way between its lower bound q_k^- and upper bound q_k^+, i.e.,

$$\frac{q_k^t - q_k^-}{q_k^+ - q_k^-} = c \qquad \frac{q_k^+ - q_k^t}{q_k^+ - q_k^-} = 1 - c. \tag{11}$$

Assuming the successive point is q_k^{t+1}, where $q_k^{t+1} \in [q_k^-, q_k^+]$, we can utilize the *golden section* to determine the optimal step size of q_k^{t+1} as

$$\begin{cases} \Delta(t, q_k^+) = c(q_k^+ - q_k^t), & \text{if } (a > b) \\ \Delta(t, q_k^-) = c(q_k^t - q_k^-), & \text{otherwise} \end{cases} \tag{12}$$

where c is the *golden fraction* with value of $(3 - \sqrt{5})/2$ and a and b represent, respectively, $|q_k^+ - q_k|$ and $|q_k - q_k^-|$. Since (12) uses constant convergence step to its ideal value, to improve the GA's efficiency and avoid unimportant search region in the early stage of evolution processing, we then incorporate evolution time t in (12), i.e.,

$$\Delta(t, q_k^+) = c(q_k^+ - q_k^t)\left(1 - r^{(1 - t/(\alpha T))}\right) \tag{13}$$

$$\Delta(t, q_k^-) = c(q_k^t - q_k^-)\left(1 - r^{(1 - t/(\alpha T))}\right) \tag{14}$$

where

r random variable distributed on the unit interval [0, 1];

α lies in [1, 1.5];

T total number of iterations.

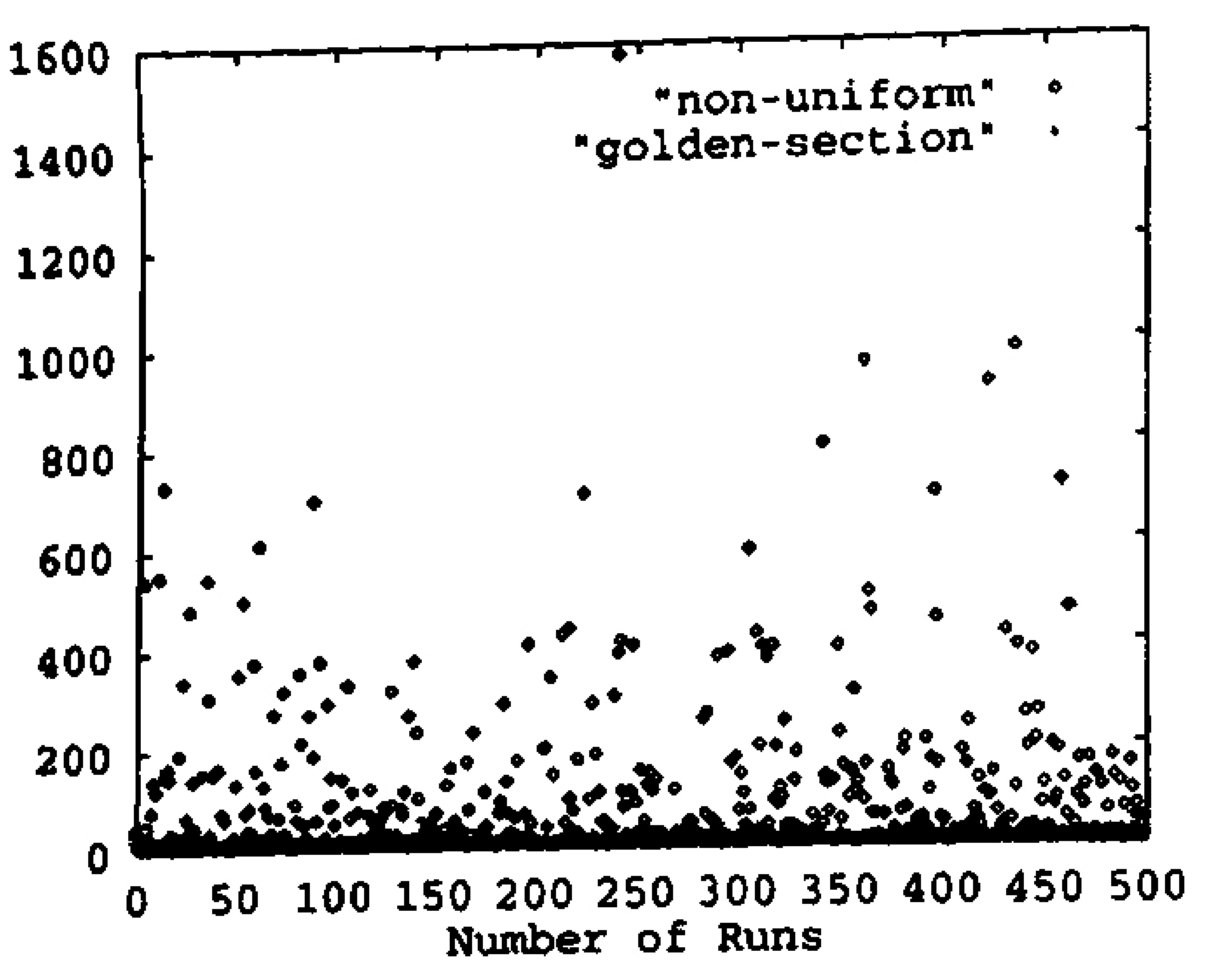

Fig. 4. Comparison of mutation schemes. Nonuniform mutation (as represented by diamonds): mean SQE error 107.2; our mutation scheme (as represented by crosses): mean SQE error 13.4. Most errors using the golden section mutation scheme are close to zero while most errors for the nonuniform mutations scheme are much higher up to 1000.

The essence of our mutation scheme, termed ***golden section***, lies in integrating (12)–(14) together and in each generation (or iteration) the scheme stochastically chooses one of them to determine the position of the new current point. The random determination of step size allows discontinuous jumps in the parameter interval, and then golden section is used to control the search direction. This ultimately makes the GA converge more accurately to a value arbitrarily close to the optimal solution. Additionally, the proposed mutation scheme requires insignificant computational time.

A comparison between the proposed mutation scheme and nonuniform mutation [22], reportedly the most effective mutation for nonlinear optimization, demonstrated the superiority of the proposed scheme in the sense of convergence as shown in Fig. 4. The figure indicates that with our mutation method, the errors generated from 500 runs all lie close to the bottom line.

C. Crossover

Crossover produces new points in the search space. The initial population forms a basis of the convex space $\mathcal{S}$ and a new individual in population is generated via (1) in Section II.

Let $\mathbf{q}_i^t$ and $\mathbf{q}_{i+1}^t$ be two individuals from population N at generation t. They satisfy

$$\mathbf{q}_i = \left\{q_j \in \left[q_j^-, q_j^+\right] \subset \mathcal{S}\right\}, \quad i = 1, \ldots, N \quad j = 1, \ldots, n \tag{15}$$

where N denotes the population size, and n the number of parameters (or chromosome length). Following (15), a new individual $\mathbf{q}_i$ in generation $t+1$ can be expressed as a linear combination of two arbitrarily selected individuals from the previous generation t, that is

$$\begin{cases} \mathbf{q}_i^{t+1} = (1-\alpha\rho_i)\mathbf{q}_i^t + \alpha\rho_i\mathbf{q}_{i+1}^t \\ \mathbf{q}_{i+1}^{t+1} = (1-\alpha\rho_{i+1})\mathbf{q}_{i+1}^t + \alpha\rho_{i+1}\mathbf{q}_i^t \end{cases} \tag{16}$$

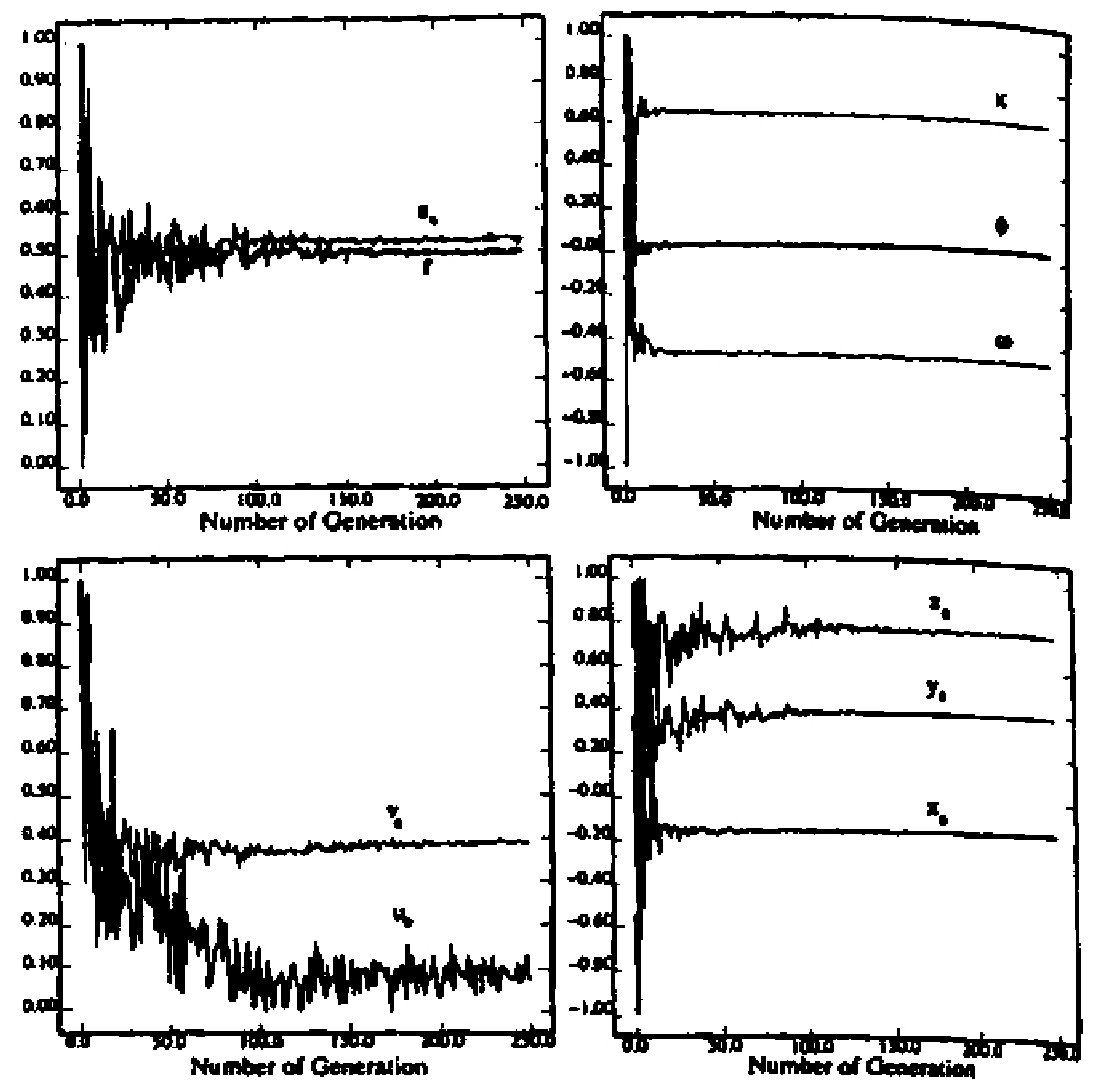

Fig. 5. Convergence performance of calibrating camera parameters; y-axis in all figures represents scaled camera parameters.

where α ranges within $[0, 1]$. ρ is a *bias factor* that increases the contribution from the dominating individual with a better fitness at current stage. Assuming nonnegative fitness function, ρ can be determined from the following equations:

$$\rho_i = \begin{cases} \xi_i, & \text{if } \xi_i < 1 \\ 1, & \text{if } \xi_i \geq 1 \end{cases} \tag{17}$$

$$\xi_i = \frac{f\left(\mathbf{q}_i^t\right)}{f\left(\mathbf{q}_{i+1}^t\right)}, \qquad \xi_{i+1} = \frac{f\left(\mathbf{q}_{i+1}^t\right)}{f\left(\mathbf{q}_i^t\right)} \tag{18}$$

where $f(*)$ denotes the GA's fitness function defined in (8).

V. Convergence and Time Complexity

As explained in the previous section, our genetic operators provide GAs with a richer population and more exploration to avoid unfavorable local minima in the early stages. And later, our GA operators gradually reduce the number of such minima qualifying for frequent visits and the attention finally shifts more to smaller refinements in the quality of solution. Fig. 5 plots convergence performance for all camera parameters as GA's evolution proceeds. It shows that after approximately 100 generations, all estimated parameters can simultaneously in parallel converge to a well stable solution. We obtained similar results in all our test cases described in next section. Different from the conventional optimization approach, GA searches a fit parameter set in the uncertain parameter space and moves toward the global optimum by gradually reducing the chance of reproducing unfit parameter sets.

Our selection mechanism consists of two procedures which are 1) roulette wheel proportionate selection and 2) linear ranking selected individual. The proportionate selection takes time approximately $(N \lg N)$ and while linear ranking needs time of about $(\lg N + \lg(\ln N))$ [23], where N denotes the population size. Let k be the number of control points on

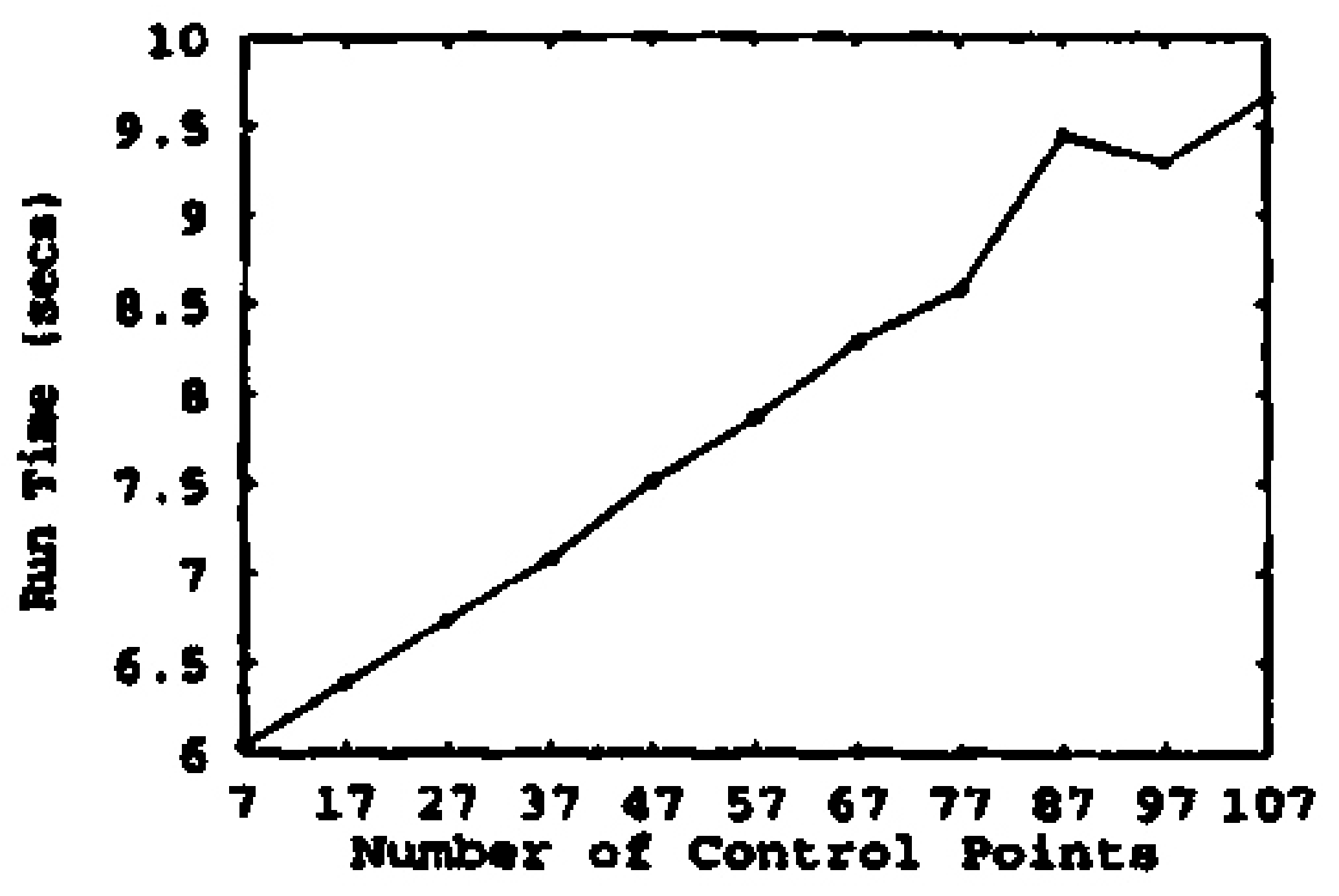

Fig. 6. The average run time (in seconds) as a function of calibration points.

calibration pattern, each evaluation then requires linear time of k. Because our mutation and crossover only involve simple arithmetics without heavy iterations, the time it takes in this procedure is comparatively negligible. Assuming that the GA converges after m generations, the time complexity t required to accomplish a calibration processing can be approximated as

$$t = m(kN \lg N + \lg N + \lg(\ln N)). \tag{19}$$

Fig. 6 offers an intuitive view of the average run time (wall time) as a function of calibration points in 300 MHz Ultra SPARC III machine, with GA population size 400 and number of generations 100. The algorithm can be further parallelized in a straightforward manner. If we simply partition the population such that each processor performs an approximately equal size of the subpopulation, the use of p processor can yield a speedup of approximately p times.The following protocol was followed to generate synthetic data.

1) Control points were randomly generated from the three visible planes of a 58 × 58 × 58 hypothetical cube. To study the performance with different numbers of control points, we selected seven (seven visible cubic corners), 47, 107 points from the cube respectively.
2) Camera parameters used to generate control points serve as absolute reference-ground truth.
3) Noise was added to the image coordinates of control points. The noise is Gaussian and independent, with a mean of zero and standard deviation of σ ranging from zero to three pixels.

VI. Experimental Results

This section describes experiments performed with synthetic and real images to evaluate the performance of our approach in terms of its accuracy and robustness under

1) varying amounts of image noise;
2) different numbers of control points;
3) different ranges of parameter bounds.

Furthermore, we describe results from a comparative performance study of our approach with that of Tsai's calibration algorithm [2]. Tsai's algorithm presented a direct solution by decoupling the 11 camera parameters into two groups; each group is solved for separately in two stages. It has since become one of the most popular camera calibration techniques in computer vision.

TABLE I
Camera Parameter Ground Truth and Bounds

Notation	Ground Truth	Parameter Bound
f	25.0	20, 40
s_x	144.0	110, 160
s_y	144.0	110, 160
u_0	256.0	200, 300
v_0	192.0	170, 230
t_x	-38.0	-80, 50
t_y	35.0	-80, 50
t_z	1210.0	900, 1400
ω	-1.90146	$-\pi$, π
ϕ	0.20916	$-\pi$, π
κ	0.15152	$-\pi$, π

A. Simulation with Synthetic Data

We generated 200 independently perturbed sets of control points for each noise level so that an accurate ensemble average of the results could be obtained. For each data set our GA was executed ten times using different initial populations generated by different random seeds. To ensure fair comparison, GA parameters were identical in all test cases.

We assess the calibration accuracy by measuring the value of both parameter errors and pixel errors. Throughout the following discussions, we defined pixel error as the average Euclidean distance between the pixel coordinates generated from the computed camera parameters and the ideal pixel coordinates. The error of individual camera parameter (camera error) is the averages Euclidean distance between the estimated camera parameter and its ground truth. For rotation matrix, the estimated error of rotation matrix is the Euclidean norm between the estimated matrix and its ground truth values.

We first investigated how image noise and control points affect the performance of our approach. For this study, the initial camera parameter bounds and their ground truth are given in Table I.

Fig. 7 plots the pixel errors and camera errors versus the number of points participating in the calibration and the amount of image noise. Table II briefly summarizes the estimation accuracy which is defined as the ratio of an estimated camera parameter and the corresponding ground truth camera parameter, where two examples of control points 7 and 107 under noise level (σ) 0.0 and 3.0 are given.

Several important observations can be made from these results. *Firstly*, the pixel error is substantially small (less than 4 pixels given $\sigma = 3$) and increases linearly with the image perturbation. Contrary to conventional techniques, our method shows no improvement in pixel errors by use of more redundant control points. Although for a few specific camera parameters, the result shows more control points may enhance the estimation accuracy, most of camera parameters such as $t_x, t_y, u_0, v_0, s_x, s_y$ show the exact opposite. Furthermore, no improvement can either be achieved in camera errors with more control points. More redundant control points may lead to the deterioration of GA's convergence since