Neuro-Calibration of a Camera using Particle Swarm Optimization

Sanjeev Kumar
Department of Maths & Comp. Science
University of Udine, Via Della Scienze 206
Udine-33100, Italy
Email: sanjeev.kumar@dimi.uniud.it

Balasubramanian Raman
Deptartment of Mathematics
IIT Roorkee
Roorkee-247667, India
Email: balarfma@ieee.org

Jonathan Wu
Department of Electrical and Comp. Eng.
University of Windsor
Windsor, Ontario, ON, N9B 3P4 Canada
Email: jwu@uwindsor.ca

***Abstract*—In this paper, a particle swarm optimization (PSO) based camera calibration approach is presented to determine the external and internal calibration parameters from the knowledge of a given set of points in object space. First, the image formation model for a pinhole camera is formulated in terms of a feed-forward neural network (NN) and then this neural network is trained using particle swarm optimization. The effect of noise and number of control points are studied in the estimation of calibration parameters. Results from our extensive study are presented to demonstrate the excellent performance of the proposed technique in terms of convergence, accuracy, and robustness.**

***Keywords*-Camera Calibration; Gaussian Noise; Neural Network; Particle Swarm Optimization (PSO); Perspective Projection.**

I. Introduction

In computer vision, camera calibration is a technique that tries to know how a camera projects a 3D object onto a 2D image plane. In other words, it is a procedure to compute the intrinsic and extrinsic parameters of the camera. This process is necessary in applications where metric information of the environment must be derived from images. Some of these applications are: making maps of the camera environment, tracking of objects, reconstruction of 3D objects and so on. Also, if the camera is installed on a moving robot, then its position with respect to objects around it can be known. This allows a robot to move in its environment avoiding obstacles, heading to a specific object, or making the definition of the best trajectory to reach its destination easily.

The existing camera calibration techniques can be broadly classified into linear [1], [2] and non-linear [3], [4] approaches. In linear approaches, due to lack of consideration of lens distortion, its calibration accuracy is difficult to meet the requirements of industrial machine vision application [5]. On the other hand, nonlinear approaches offer a more accurate and robust solution [6]. The drawbacks of nonlinear approaches are computationally expensive and require good initial estimates. The other common strategy for camera calibration is a two-step method given in [7]. The first step generates an approximate solution using a linear approach, and this solution is improved by using a nonlinear iterative process in the second step.

Apart from these approaches an alternative manner based on soft computing techniques has been developed to solve calibration problem. This problem has been solved by using binary coded genetic algorithm [8] as well as real coded genetic algorithm [9], [10] in a robust manner. Some researchers have solved the calibration problem using artificial neural network [11], [12]. However, the main drawback to employ neural network to solve this problem is estimation of good initial values of weight vectors of network [13], [14]. An effort has been made to overcome this drawback by employing neural network and genetic algorithm together [15]. In the same manner, particle swarm optimization based approaches are introduced to calibrate the pinhole camera model. Recently, a PSO based approach [16] for self-calibration has been given by minimizing the cost function generated from Kruppa's equations. In [17], an optimization based approach using PSO has been given for nonlinear camera calibration.

In this paper, camera calibration problem is formulated in terms of a neural network and then this network is trained by a particle swarm optimization based learning algorithm. Most of the existing neural network based calibration techniques use neural networks either for learning the mapping from the 3D world to 2D images without specifying the calibration parameters [14] or as an additional stage to improve the performance of other existing techniques [15]. This technique is able to specify the camera intrinsic and extrinsic parameters since knowing these parameters is useful for many vision tasks such as stereo reconstruction. The proposed PSO based neural network training algorithm provides many advantages such as no need for any specific initial estimates, high probability for converging to global minima and easy realization when compared to standard backpropagation learning algorithm.

The rest of the paper is organized as follows: a description for the image formation process of pinhole camera is given in section II. In section III, formulation of calibration problem in terms of neural network is given. Section IV provides the given PSO-based training algorithm. Experimental Results are given in section V. The concluding remarks are given in section VI.

II. Image Formation Model

The camera model used in this paper is a well known pinhole camera model based on perspective projection (see Fig. 1). Let $M(x_m, y_m, z_m)$ be a point in 3D object surface with respect to world coordinate system $oxyz$ and $OXYZ$ is the camera coordinate system with O as an optical center of the camera. O_iXY is the retinal coordinate system of the camera where O_i is the intersection of optical axis Z and the image plane of the camera. (X_u, Y_u) is the retinal coordinates of the camera in the ideal keyhole model and (X_d, Y_d) is the actual retinal coordinates, considering the lens distortion. The distance between optical center and image plane (focal length) is f.

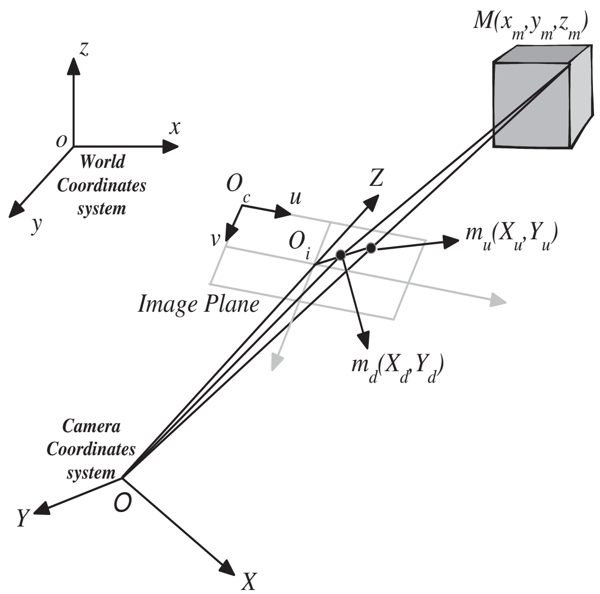

Figure 1. Image formation model using pinhole camera

The steps in which 3D world coordinates have been transformed into pixel coordinates (u, v) are given as follows:

1) Transform 3D world coordinates into camera coordinates:
$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = R. \begin{pmatrix} x_m \\ y_m \\ z_m \end{pmatrix} + T \qquad (1)$$
where R is a 3×3 rotation matrix representing the orientation of camera corresponding to the world coordinate system. It can be expressed completely in terms of Yaw, Pitch and Roll (YPR) angles $(\alpha,\ \beta,\ \gamma)$. T is the position of the camera corresponding to world coordinate system.
2) Transform 3D camera coordinates (X, Y, Z) into ideal retinal coordinates (X_u, Y_u) using perspective projection equations:
$$X_u = f\frac{X}{Z}, \qquad Y_u = f\frac{Y}{Z} \qquad (2)$$
3) Transform ideal retinal coordinates (X_u, Y_u) into actual retinal coordinates (X_d, Y_d) by considering lens radial distortion coefficient k:
$$X_d = X_u(1 + kr^2)^{-1} \qquad Y_d = Y_u(1 + kr^2)^{-1} \qquad (3)$$
where r is the distance between image center O_i and actual retinal coordinates.
4) Transform actual retinal coordinates (X_d, Y_d) into pixel coordinates (u, v):
$$u = X_dN_x + u_0 \qquad v = Y_dN_y + v_0 \qquad (4)$$
where (u_0, v_0) represents the pixel coordinates of image center O_i; N_x and N_y express the number of pixels that contain in the unit distance of image plane along X and Y axis respectively.

On combining above four steps, the image formation process of a 3D point $M\ (x_m, y_m, z_m)$ can be written as
$$S \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = P \begin{pmatrix} x_m \\ y_m \\ z_m \\ 1 \end{pmatrix} \qquad (5)$$
where S is a scaling factor and the matrix P is decomposed into two matrices: $P = A \quad D$, where $D = [R \qquad T]$ and A is a 3×3 matrix containing the intrinsic parameters of camera $(f, k, N_x, N_y, u_0, v_0)$.

The matrix P can be written as follows:
$$P = \begin{pmatrix} p_1^T & p_{14} \\ p_2^T & p_{24} \\ p_3^T & p_{34} \end{pmatrix} \qquad (6)$$
where $(p_i^T \quad p_{i4})$; $(i = 1, 2, 3)$, represent the rows of matrix P. Using equations (5) and (6), the pixel coordinates $(u,\ v)$ can be represented as
$$u = \frac{p_1^TM + p_{14}}{p_3^TM + p_{34}}, \qquad v = \frac{p_2^TM + p_{24}}{p_3^TM + p_{34}} \qquad (7)$$
Equation (7) represents the transformation of 3D world coordinates (x, y, z) into pixel coordinates (u, v) for a single camera. This process is irreversible, i.e., it can not determine 3D position of point w uniquely from pixel coordinates (u, v). However, a point $M(x_m, y_m, z_m)$ in 3D world can be determined uniquely from its stereo images $w_l(u_l, v_l)$ and $w_r(u_r, v_r)$ taken by two cameras (left and right) or from a single camera in two different positions.

III. Image Model to Neural-Network Formulation

Let q be a vector consisting required unknown intrinsic and extrinsic parameters of the respective camera
$$q = (u_0\ v_0\ N_x\ N_y\ f\ k\ \alpha\ \beta\ \gamma\ T_x\ T_y\ T_z)^T \qquad (8)$$
For notational convenience, we write q as $q = (q_1,\ q_2, ..., q_{12})^T$. With the earlier described camera model,

the calibration problem can be stated as follows: Given sufficient number of control points N whose world coordinates $M_i(x_{mi}, y_{mi}, z_{mi})$ are known with a high precision, as well as their corresponding observed pixel positions (u_i, v_i) in image coordinates, the problem is to estimate the optimal solution of 12 camera parameters given by equation (8), which minimize the following objective function

$$E = \sum_{i=1}^{N} \left[\left(\frac{p_1^T M_i + p_{14}}{p_3^T M_i + p_{34}} - u_i \right)^2 + \left(\frac{p_2^T M_i + p_{24}}{p_3^T M_i + p_{34}} - v_i \right)^2 \right] \quad (9)$$

Our interest is to employ a neural network not only to learn the mapping from 3D points to 2D pixel points, which minimizes the error in (9), but also to extract the projection matrix and camera parameters. Therefore, the network structure is laid out accordingly. The NN is a feed-forward network and composed by three layers. The input has three neurons corresponding to the three coordinates (x_m, y_m, z_m) of a 3D point. The number of output units is three, and the hidden layer consists of four neurons. The weight matrix of the hidden layer is denoted by $W^[1]$, and it is assumed to correspond to the extrinsic parameters matrix D. The weight matrix of the output layer is denoted as $W^[2]$ and corresponds to the intrinsic parameters, or matrix A. For any input pattern i the input vector is formed as $J_i = (x_{mi}, y_{mi}, z_{mi})^T$, and the outputs are (O_{i1}, O_{i2}, O_{i3}). The error measure here would be

$$E = \sum_{i=1}^{N} \left[\left(\frac{O_{1i}}{O_{3i}} - u_i \right)^2 + \left(\frac{O_{2i}}{O_{3i}} - v_i \right)^2 \right] \quad (10)$$

where $O_{ji} = p_j^T M_i + p_{j4};\ j = 1, 2, 3$ and (u_i, v_i) are the desired pixel coordinates of the input point patterns. This last equation is not in a handy form that can be used by a backpropoagation network due to the presence of the two ratios in terms of the network outputs. To tackle this problem, we have used a PSO based training for neural network.

IV. PSO BASED NETWORK TRAINING

A. PSO: Basic Introduction

Particle swarm optimization (PSO) is an evolutionary computational model which maintains a swarm of candidate solutions, referred as particles [18]. Particles, as flown through the search spaces, are attracted towards the best solution found by the neighboring particles and by these particle. Each particle has a position vector and velocity vector. The position vector and velocity vector of the i^{th} particle in the n-dimensional search spaces can be represented as $X_i(t) = (x_{i1}(t), x_{i2}(t), x_{i3}(t), ..., x_{in}(t))$ and $V_i(t) = (v_{i1}(t), v_{i2}(t), v_{i3}(t), ..., v_{in}(t))$ respectively. Each particle also keeps track of its best position value, and the best position encountered by each particle is represented as $P_i(t) = (p_{i1}(t), p_{i2}(t), p_{i3}(t), ..., p_{in}(t))$. The best particle among all the particles found so far at time t is represented as $P_b(t) = (p_{b1}(t), p_{b2}(t), p_{b3}(t), ..., p_{bn}(t))$. The PSO algorithm could be performed by the following equations:

$$\begin{aligned} v_{in}(t+1) = k * v_{in}(t) + c_1 * r_{in}(t) * (p_{in}(t) \\ -x_{in}(t)) + c_2 * r_{2n} * (p_{bn}(t) - x_{in}(t)), \end{aligned} \quad (11)$$

and

$$x_{in}(t+1) = x_{in}(t) + v_{in}(t+1), \quad (12)$$

where $r_{1n}(t)$ and $r_{2n}(t) \in U(0,1)$ are uniformly distributed random numbers between 0 and 1, c_1 and c_2 are acceleration constants, and k denotes the inertia weight. The inertia weight k is a user-defined parameter that controls, with c_1 and c_2. The terms $c_1 * r_{in}(t) * (p_{in}(t) - x_{in}(t))$ and $c_2 * r_{2n} * (p_{gn}(t) - x_{in}(t))$ are the cognitive and social components, respectively. The particle's new velocity is updated by using (12) based on its previous velocity, distances of its current positions from its own best historical position and the collaborative effect of particles. Finally, the new position of the particle is updated using (12) to get the optimal solution.

B. Training Algorithm

Most of the work involving the evolution of neural network using evolutionary computation has focused on the network weights and topological structure such as the weights and/or topological structure are encoded as a chromosome in GA. The selection of fitness function depends on the research goals. In network training, PSO can be used as a replacement of the back-propagation learning algorithm to get a faster convergence and avoid initial estimates problem.

Our network model contains three different layers as input, hidden and output. The code is simply set up to work with layers separately and the particle swarm treats the entire set of matrices as one long vector having 12 elements corresponding to the network weights. The representation of the connection weight matrix of size 12×2 of the i^{th} node is as follows

$$W_i = (W_i^{[1]}, W_i^{[2]}),$$

where $W_i^{[1]}$ and $W_i^{[2]}$ represent the connection weight matrices of the i^{th} node between the input layer and the hidden layer, and that between the hidden layer and the output layer, respectively. Moreover, the vector of the position of the previous best fitness value of any particle is represented by

$$P_i = (P_i^{[1]}, P_i^{[2]}),$$

where $P_i^{[1]}$ and $P_i^{[2]}$ represent the position of the previous best fitness value of the i^{th} particle, between the input layer and the hidden layer, and that between the hidden layer and the output layer, respectively.

The index of the best particle among all the particles in the population is represented by the symbol b. So the best matrix is represented by

$$P_b = (P_b^{[1]}, P_b^{[2]})$$

where $P_b^{[1]}$ and $P_b^{[2]}$ represent the position of the best particle among all the particles, between the input layer and the hidden layer, and that between the hidden layer and the output layer, respectively.

The velocity of the particle i is denoted by

$$V_i = (V_i^{[1]}, V_i^{[2]})$$

If m and n represent the index of matrix row and column, respectively, then the new velocity and connection weights can be updated as

$$V_{i+1}^j = k * V_i^j(m,n) + c_1 * r_1^j * (P_i^j(m,n) \quad (13)$$
$$-W_i^j(m,n)) + c_2 * r_2^j * (P_b^j(m,n) - W_i^j(m,n)),$$

and

$$W_{i+1}^j = W_i^j + V_{i+1}^j \quad (14)$$

where $j = 1, 2$; $m = 1, 2, \ldots, M_j$; $n = 1, 2, \ldots, N_j$. M_j and N_j are the row and column sizes of the weight matrices W, position matrix P, and the velocity matrix V. The new velocity of the particle based on its previous velocity and the distances of its current position is computed using (14) from the best experiences both in its own and as a group. In the context of social behavior, the cognition part $c_1 * r_1^j * (P_i^j(m,n) - W_i^j(m,n))$ represents the private thinking of the particle itself whilst the social part $c_2 * r_2^j * (P_b^j(m,n) - W_i^j(m,n))$ denotes the collaboration among the particles as a group. Finally, the new position (or weights) according to the new velocity is updated according to (14).

The fitness of the i^{th} particle is expressed in term of an output mean squared error of the neural networks as

$$\text{Fitness} = \sum(\text{Actual value} - \text{Network predicted value})^2 \quad (15)$$

Once the fitness function meet the convergence requirements, the values of parameters $q = (q_1, q_2, \ldots, q_{12})$ can be estimated from the final weights W to accomplish the calibration task.

V. Experimental Results

This section describes experimental study performed on synthetic data to evaluate the performance of proposed approach in terms of accuracy and robustness using different number of control points and under varying amount of image noise. Synthetic data has been generated in the following manners:

1) Control points were randomly generated from the three visible planes of a $30 \times 30 \times 30$ hypothetical cube. To study the performance with different numbers of control points, we selected 7 (visible corners of the cube) and 57 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 the control points. The noise is Gaussian with zero mean and standard deviation ranging from zero to four pixels.

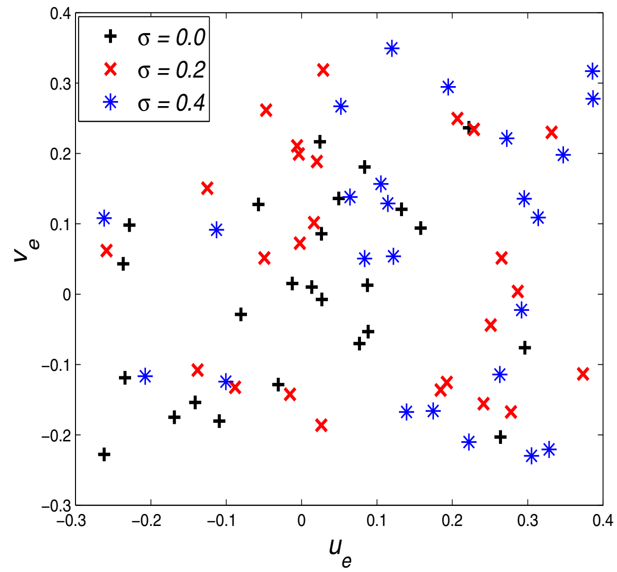

Figure 2. Reprojection errors (in pixels) for 25 control points in the presence of various amount of Noise.

Table I
CAMERA PARAMETERS (GROUND-TRUTH)

Parameters	f	N_x	N_y	u_0	v_0	k
Ground Truth	10	80	120	20	40	0
Parameters	T_x	T_y	T_z	α	β	γ
Ground Truth	35	10	100	$\pi/12$	$\pi/5$	0

The ground truth values for various calibration parameters used in the generation of synthetic data are shown in Table I. The stopping criterion for the training of neural network is fixed as 10^{-4} or 1000 epochs whichever is earlier. The calibration accuracy for every calibration parameter has been measured as follows:

$$\text{Accuracy}(\%) = \left(\frac{\text{Estimated value}}{\text{Ground-truth}}\right) \times 100 \quad (16)$$

The robustness of the proposed calibration algorithm is checked by adding the Gaussian noise in the control points.

The accuracy and robustness of the proposed method, in presence of different noise levels, have been shown in

Table II
RESULTS FOR THE ESTIMATED CALIBRATION ACCURACY IN VARIOUS CAMERA PARAMETERS IN PRESENCE OF GAUSSIAN NOISE

Parameters	Estimated Accuracies (%)					
	$\sigma = 0.0$		$\sigma = 2.0$		$\sigma = 4.0$	
	N=57	N=7	N=57	N=7	N=57	N=7
f	95.8	95.9	95.7	95.4	95.8	95.6
N_x	96.3	96.3	96.4	96.2	95.9	96.2
N_y	97.2	97.0	97.4	97.5	97.7	96.6
u_0	97.4	97.2	97.5	97.1	96.8	96.8
v_0	98.1	98.4	97.7	97.9	98.1	98.0
k	99.2	98.6	99.5	99.8	99.1	99.4
T_x	93.7	93.8	94.2	93.4	92.8	92.8
T_y	94.9	94.8	95.1	95.6	94.2	94.4
T_z	95.6	95.0	94.8	94.7	95.1	95.2
α	97.3	97.4	97.1	97.2	96.9	96.4
β	95.9	95.4	94.7	94.6	95.1	95.0
γ	96.6	96.1	96.4	96.5	95.7	95.5

Table II. Several important observations can be made from these results. The reprojection errors for 25 control points in the presence of Gaussian noise having zero mean and variance $\sigma = 0.0,\ 0.2$ and 0.4 are shown in Fig. 2. Contrary to conventional techniques, our method shows very little improvement in calibration error by use of more control points. Although for some specific camera parameters, the results show that more control points may enhance the estimation accuracy. Furthermore, no specific improvement can be achieved in camera errors with more control points. It is found that the presence of error does not effect the accuracy of the proposed algorithm. In most of the cases, it is noticed that the accuracy is not decreasing by increasing the amount of Gaussian noise in control points as well as number of control points. Thus, the proposed algorithm presents very robust results with sufficient accuracy.

VI. CONCLUSIONS

In this paper, a PSO based calibration algorithm has been presented to solve the camera calibration problem. The calibration problem is formulated in terms of a feed-forward neural network containing three layers. A PSO based training algorithm has been employed in the neural network training. The robustness and accuracy of proposed method have been shown by simulation studies. From the simulation results, it is concluded that the accuracy in solution does not depend on the number of control points. The proposed method shows the robustness in the presence of various amount of Gaussian pixel noise. Further, accuracy can be increased up to some extent by modifying the stopping criterion of network.

ACKNOWLEDGEMENTS

Balasubramanian Raman gratefully acknowledges the financial support of Department of Science and Technology(DST), India under his BOYSCAST fellowship to carry out this research work at Dept. of Electrical and Computer Engg., University of Windsor, Canada.

REFERENCES

[1] W. Faig, Calibration of close-range photogrammetry systems: Mathematical formulation, Photogramm. Eng. Remote Sensing, vol. 41, 1975, pp. 1479-1486.

[2] L. Paquette, R. Stampfler and W.A. Devis and T.M. Caelli, A new camera calibration method for robotic vision, in Proceedings SPIE: Closed Range Photogrammetry Meets Machine Vision, 1990, pp. 656-663, Zurich, Switzerland.

[3] D.B. Gennery, Stereo-camera calibration, in Proceedings of Image Understanding Workshop, 1979, pp. 101-108, Menlo Park, CA.

[4] A. Okamoto, Orientation and construction of models: The orientation problem in close-range photogrammetry, Photogramm. Eng. Remote Sensing, vol. 5, 1981, pp. 1437-1454.

[5] I. Sobel, On calibrating computer controlled cameras for perceiving 3D scenes, IEEE Transcations on Artificial Intelligence, vol. 5, 1974, pp. 1437-1454.

[6] K.W. Wong, Mathematical formulation and digital analysis in closerange photogrammetry, Photogramm. Eng. Remote Sensing, vol. 41(11), 1975, pp. 1355-1373.

[7] J. Weng, P. Cohen and M. Herniou, Camera calibration with distortion models and accuracy evaluation, IEEE Transcations on Pattern Analysis and Machine Intelligence, vol. 10, 1992, pp. 965-980.

[8] S. Hati and S. Sengupta, Robust camera parameter estimation using genetic algorithm, Pattern Recognition Letters, vol. 22(3), 2001, pp. 289–298.

[9] S. Kumar, M. Thakur, R. Balasubramanian, N. Sukavanam, Stereo camera calibration using real coded genetic algorithm, IEEE Region 10 Conference (TENCON), 2008, 1–5, Hyderabad, India.

[10] Q. Ji and Y. Zhang, Camera calibration with genetic algorithms, IEEE Transactions on Systems, Man and Cybernatics Part A: Systems and Humans, vol. 31(2), 2001, pp. 120-130.

[11] Y. Xing, Q. Liu, J. Sun and L. Hu, Analyzing and improving of neural networks used in stereo calibration, in Proceedings of IEEE ICNC-2007, Haikou, China, 1:745–749, August 2007.

[12] M. Ahmed, E. Hemayed and A. Farag, Neurocalibration: a neural network that can tell camera calibration parameters, Proceedings of IEEE International Conference on Computer Vision, September 1999.

[13] J. Jun and C. Kim, Robust Camera Calibration using Neural Network, IEEE TENCON, 694-697, September 1999.

[14] Q. Memon and S. Khan, Camera Calibration and Three-dimensional world reconstruction of stereo vision using neural networks, International Journal of Systems Science, 32:1155-1159, August 2001.

[15] N. Sukavanam, R. Balasubramanian, M. Thakur, S. Kumar, Camera calibration based on Hybridization of Genetic Algorithm and Neural Network, International Journal of Information Processing, Volume 2, Number 3, 2008, pp. 1-11.

[16] J.Z. Tao, W. Wenhuan, W. Min, Camera Autocalibration from Kruppa's equations using particle Swarm Optimization, IEEE Int. Conf. on Computer Science and Software Engineering, 2008, Volume 1, pp. 1032–1034, China.

[17] D. Wang, Y. Tu, T. Zhang, Research on the Application of PSO Algorithm in Non-linear Camera Calibration, World Congress on intelligent Control and Automation, 2008, Chongqing, China.

[18] J. Kennedy, R. Eberhart, Particle Swarm Optimization, IEEE Int. Conf. on Neural networks, 1995, 1942-1948.