


# **Localization for Legged Robot with Single Low Resolution Camera using Genetic Algorithm**



**TONG, Fung Ling**

A Thesis Submitted in Partial Fulfillment  
of the Requirements for the Degree of  
Master of Philosophy  
in  
Electronic Engineering

© The Chinese University of Hong Kong  
September 2007

The Chinese University of Hong Kong holds the copyright of this thesis. Any persons(s) intending to use a part or whole of the materials in the thesis in a proposed publication must seek copyright release from the Dean of the Graduate School.

Localization for Eggbot Robot with ZigBee  
Resolution Center using Genetic Algorithm



A Thesis Submitted in Partial Fulfillment  
of the Requirements for the Degree of  
Master of Philosophy  
in  
Electronic Engineering

© The Chinese University of Hong Kong  
December 2007

The Chinese University of Hong Kong is a registered trademark of the Chinese University of Hong Kong.  
All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or by any information storage and retrieval system, without the prior written permission of the Chinese University of Hong Kong.

Abstract of thesis entitled:  
**Localization for Legged Robot with Single Low Resolution  
Camera using Genetic Algorithm**

Submitted by **TONG Fung Ling**  
for the degree of **Master of Philosophy**  
in **Electronic Engineering**  
at **The Chinese University of Hong Kong**  
in **September 2007.**

*The simultaneous localization and mapping technique is an important requirement in the development of autonomous robot. Many localization algorithms for wheeled robots using various sensors have been proposed. In this thesis, we present a vision-based localization algorithm for a small home-use robot pet (legged robot) which is equipped with a low-resolution camera as the only sensor for localization. Challenges of vision-based localization for legged robots include: 1) leg slippages are common in legged robot, which lead to unpredicted and unmodeled motion errors, 2) as the sensor data is fluctuated due to the oscillated walking motion of legged robot, the high degree of freedom of legged robot increases the complexity of the localization problem and 3) camera has limited field of view and image points are lack of depth information. In the proposed algorithm, the localization for high-dimensional movement robot is modeled as an optimization. The robot state (position and orientation) is obtained by optimizing the formulated objective function using genetic algorithm (GA). Based on the vision-based localization problem, details of the employment of the GA are presented. Besides, approaches that aim to 1) increase the efficiency of the search and 2) weaken the influence of noisy feature points on the localization results are proposed. Results from simulations show that the proposed algorithm is able to localize the legged robot accurately and efficiently even though the input feature points involve high level of noise.*

## 摘要

同時定位與地圖創建技術在發展全自主機器人的領域上是一項重要的條件。研究人員對於使用不同傳感器的輪式機器人, 已提出大量不同的定位演算法。在這篇論文中, 我們針對家用的寵物機器人(腿式機器人), 提出一個基於視覺的定位演算法。這種機器人只裝備一臺低分辨率的攝影機, 作為唯一的定位用傳感器。研究腿式機器人的視覺定位方法之挑戰在於: 1) 腿式機器人在步行時滑動是很普遍的, 這導致機器人的移動產生不能預料和不能建模的誤差, 2) 因為機器人擺動的走動方式令傳感器和傳感器接收的資料振盪, 所以腿式機器人的高自由度移動增加了定位問題的複雜性, 以及 3) 相對於其它傳感器, 攝影機的視野有限, 而且圖像缺乏距離的資料。在我們提出的定位演算法, 高自由度移動機器人的定位被建模為一個最優化程序, 通過使用遺傳演算法來優化目標函數式, 因而獲得機器人的位置和方向。我們將會提出針對腿式機器人視覺定位問題之遺傳演算法的運用細節。此外, 我們提出方法以提高優化程序的效率和減弱噪聲圖像對定位結果的影響。模擬實驗結果顯示, 即使使用噪聲圖像特徵, 本論文提出的視覺定位演算法仍能對腿式機器人作出高效率 and 準確的定位。

## ***Acknowledgement***

I would like to express my gratitude to my supervisor, Prof. MAX Q.-H. MENG. Thank him for giving me a chance to experience the world of research. This experience is valuable to me. Also, I would like to thank him for his patient guidance and valuable opinions. Deepest gratitude is also expressed to all beloved family for take caring me all the time. Special thanks to Kin for his understanding, encouragements and help throughout my studies, especially during the thesis writing period.

# Table of Contents

<b>Abstract</b> .....	<b><i>i</i></b>
<b>摘要</b> .....	<b><i>ii</i></b>
<b>Acknowledgement</b> .....	<b><i>iii</i></b>
<b>Table of Contents</b> .....	<b><i>iv</i></b>
<b>List of Figures</b> .....	<b><i>vii</i></b>
<b>List of Tables</b> .....	<b><i>x</i></b>
<b>Chapter 1 – Introduction</b> .....	<b><i>1</i></b>
<b>Chapter 2 – State of the art in Vision-based Localization</b> .....	<b><i>6</i></b>
<b>2.1 Extended Kalman Filter-based Localization</b> .....	<b><i>6</i></b>
2.1.1 Overview of the EKF algorithm.....	<b><i>6</i></b>
2.1.2 Process of the EKF-based localization algorithm.....	<b><i>8</i></b>
2.1.3 Recent EKF-based vision-based localization algorithms...	<b><i>10</i></b>
2.1.4 Advantages of the EKF-based localization algorithms.....	<b><i>11</i></b>
2.1.5 Disadvantages of the EKF-based localization algorithm...	<b><i>11</i></b>
<b>2.2 Monte Carlo Localization</b> .....	<b><i>12</i></b>
2.2.1 Overview of MCL.....	<b><i>12</i></b>
2.2.2 Recent MCL-based localization algorithms.....	<b><i>14</i></b>
2.2.3 Advantages of the MCL-based algorithm.....	<b><i>15</i></b>
2.2.4 Disadvantages of the MCL-based algorithm.....	<b><i>16</i></b>
<b>2.3 Summary</b> .....	<b><i>16</i></b>
<b>Chapter 3 – Vision-based Localization as an Optimization Problem</b> .....	<b><i>18</i></b>
<b>3.1 Relationship between the World, Camera and Robot Body Coordinate System</b> .....	<b><i>18</i></b>
<b>3.2 Formulation of the Vision-based Localization as an Optimization Problem</b> .....	<b><i>21</i></b>
<b>3.3 Summary</b> .....	<b><i>26</i></b>

<b>Chapter 4 – Existing Search Algorithms</b> .....	27
4.1 <i>Overview of the Existing Search Algorithms</i> .....	27
4.2 <i>Search Algorithm for the Proposed Objective Function</i> .....	28
4.3 <i>Summary</i> .....	30
<b>Chapter 5 – Proposed Vision-based Localization using Genetic Algorithm</b> .....	32
5.1 <i>Mechanism of Genetic Algorithm</i> .....	32
5.2 <i>Formation of Chromosome</i> .....	35
5.3 <i>Fitness Function</i> .....	39
5.4 <i>Mutation and Crossover</i> .....	40
5.5 <i>Selection and Stopping Criteria</i> .....	42
5.6 <i>Adaptive Search Space</i> .....	44
5.7 <i>Overall Flow of the Proposed Algorithm</i> .....	46
5.8 <i>Summary</i> .....	47
<b>Chapter 6 – Experimental Results</b> .....	48
6.1 <i>Test Robot</i> .....	48
6.2 <i>Simulator</i> .....	49
6.2.1 <i>Camera states simulation</i> .....	49
6.2.2 <i>Oscillated walking motion simulation</i> .....	50
6.2.3 <i>Input images simulation</i> .....	50
6.3 <i>Computer for simulations</i> .....	51
6.4 <i>Position and Orientation errors</i> .....	51
6.5 <i>Experiment 1 – Feature points with quantized noise</i> .....	53
6.5.1 <i>Setup</i> .....	53
6.5.2 <i>Results</i> .....	56
6.6 <i>Experiment 2 – Feature points added with Gaussian noise</i> .....	62
6.6.1 <i>Setup</i> .....	62
6.6.2 <i>Results</i> .....	62
6.7 <i>Experiment 3 – Noise reduction performance of the adaptive search space strategy</i> .....	77
6.7.1 <i>Setup</i> .....	77
6.7.2 <i>Results</i> .....	79
6.8 <i>Experiment 4 – Comparison with benchmark algorithms</i> .....	83
6.8.1 <i>Setup</i> .....	83

6.8.2 Results.....	85
6.9 Discussions.....	88
6.10 Summary.....	90
<b>Chapter 7 – Conclusion</b> .....	91
<b>References</b> .....	94



## List of Figures

2.1	Flow diagram of the Extended Kalman filter – based localization algorithm.....	10
2.2	Flow diagram of the Monte Carlo Localization algorithm. ....	14
3.1	Relationship between the world coordinate system $W$ , the camera coordinate system $C$ and the robot body coordinate system $B$ .....	19
3.2	Relationship between the landmark position vector ( $\mathbf{f}_i$ and $\mathbf{h}_i$ ) and the projected image point $\mathbf{g}_i$ . ....	23
3.3	Visualization of the goodness of a camera state ( $N_m = 2$ ).....	25
4.1	The objective function $f_{goodness}(r)$ is multimodal.....	29
5.1	Chromosome and genes.....	32
5.2	The mutation process. ....	33
5.3	The crossover process.....	34
5.4	Flow diagram of genetic algorithm. ....	35
5.5	Example of multiple expressions of a rotation in Euler angles. ....	36
5.6	Representation of a point in a unit sphere using $\alpha$ , $\beta$ and $l$ . ....	38
5.7	Example of crossover with the swapping vector = [1, 0, 0, 1, 1, 0].....	41
5.8	Example of mutation with mutating vector = [1, 0, 0, 0, 1, 0].....	42
5.9	The selection process.....	43
5.10	An example of $\Omega_p^{(i)}$ for $n = 2$ .....	45
5.11	Flow diagram of the proposed localization algorithm using genetic algorithm.....	46
6.1	Test robot: Sony AIBO ERS-7. ....	49
6.2	(a) Landmarks and actual path (Topview) and (b) magnified figure of actual path (Topview). ....	54
6.3	The actual paths of 100 trials. ....	55
6.4	Experimental results of feature points with quantized noise: (a) average position errors, (b) average orientation errors and (c) average processing time per frame. ....	57
6.5	(a) Actual paths and (b-g) estimated paths of <i>feature</i> points with quantized <i>noise</i> , $N_p = 100$ and (b) $N_m = 5$ , (c) $N_m = 6$ , (d) $N_m = 7$ , (e) $N_m = 8$ , (f) $N_m = 9$ and (g) $N_m = 10$ .....	59

6.6	(a) Actual paths and (b-g) estimated paths of <i>feature</i> points with quantized <i>noise</i> , $N_p = 200$ and (b) $N_m = 5$ , (c) $N_m = 6$ , (d) $N_m = 7$ , (e) $N_m = 8$ , (f) $N_m = 9$ and (g) $N_m = 10$ .....	60
6.7	(a) Actual paths and (b-g) estimated paths of <i>feature</i> points with quantized <i>noise</i> , $N_p = 300$ and (b) $N_m = 5$ , (c) $N_m = 6$ , (d) $N_m = 7$ , (e) $N_m = 8$ , (f) $N_m = 9$ and (g) $N_m = 10$ .....	61
6.8	Experimental results of feature points added with Gaussian noise ( $\sigma_n = 4$ pixels) : (a) average position errors, (b) average orientation errors and (c) average processing time per frame.....	63
6.9	Experimental results of feature points added with Gaussian noise ( $\sigma_n = 8$ pixels) : (a) average position errors, (b) average orientation errors and (c) average processing time per frame.....	65
6.10	Position errors for the population size (a) 100, (b) 200 and (c) 300. In each sub-figure, the standard deviation $\sigma_n$ of the added Gaussian noise is varied from 0 to 8 pixels.....	67
6.11	Orientation errors for the population size (a) 100, (b) 200 and (c) 300. In each sub-figure, the standard deviation $\sigma_n$ of the added Gaussian noise is varied from 0 to 8 pixels.....	68
6.12	(a) Actual paths and (b-g) estimated paths of feature points added with Gaussian noise ( $\sigma_n = 4$ pixels), $N_p = 100$ and (b) $N_m = 5$ , (c) $N_m = 6$ , (d) $N_m = 7$ , (e) $N_m = 8$ , (f) $N_m = 9$ and (g) $N_m = 10$ .....	71
6.13	(a) Actual paths and (b-g) estimated paths of feature points added with Gaussian noise ( $\sigma_n = 4$ pixels), $N_p = 200$ and (b) $N_m = 5$ , (c) $N_m = 6$ , (d) $N_m = 7$ , (e) $N_m = 8$ , (f) $N_m = 9$ and (g) $N_m = 10$ .....	72
6.14	(a) Actual paths and (b-g) estimated paths of feature points added with Gaussian noise ( $\sigma_n = 4$ pixels), $N_p = 300$ and (b) $N_m = 5$ , (c) $N_m = 6$ , (d) $N_m = 7$ , (e) $N_m = 8$ , (f) $N_m = 9$ and (g) $N_m = 10$ .....	73
6.15	(a) Actual paths and (b-g) estimated paths of feature points added with Gaussian noise ( $\sigma_n = 8$ pixels), $N_p = 100$ and (b) $N_m = 5$ , (c) $N_m = 6$ , (d) $N_m = 7$ , (e) $N_m = 8$ , (f) $N_m = 9$ and (g) $N_m = 10$ .....	74
6.16	(a) Actual paths and (b-g) estimated paths of feature points added with Gaussian noise ( $\sigma_n = 8$ pixels), $N_p = 200$ and (b) $N_m = 5$ , (c) $N_m = 6$ , (d) $N_m = 7$ , (e) $N_m = 8$ , (f) $N_m = 9$ and (g) $N_m = 10$ .....	75
6.17	(a) Actual paths and (b-g) estimated paths of feature points added with Gaussian noise ( $\sigma_n = 8$ pixels), $N_p = 300$ and (b) $N_m = 5$ , (c) $N_m = 6$ , (d) $N_m = 7$ , (e) $N_m = 8$ , (f) $N_m = 9$ and (g) $N_m = 10$ .....	76
6.18	Landmarks and 100 independent camera states (Topview). ....	77
6.19	Differences between actual camera states and global optimums for $N_m = 5$ and $\sigma_n = 4$ pixels (Topview).....	80

6.20	Differences between actual camera states and global optimums for $N_m = 5$ and $\sigma_n = 8$ pixels (Topview).....	80
6.21	Sorted distances between the global optimums and the actual camera states for feature points added with Gaussian noise with standard deviation $\sigma_n = 4$ pixels and 8 pixels.....	81
6.22	Average distances between the actual camera states and the global optimums, and average position errors with feature points added with Gaussian noise ( $\sigma_n = 4$ pixels).....	82
6.23	Average distances between the actual camera states and the global optimums, and average position errors with feature points added with Gaussian noise ( $\sigma_n = 8$ pixels).....	82
6.24	Localization results of the EKF-based localization algorithm (Topview).....	86
6.25	Localization results of the MCL algorithm (Topview).....	86
6.27	Position errors of localization results using the EKF-based algorithm....	87
6.28	Position errors of localization results using the MCL algorithm.....	87

## List of Tables

4.1	Limitations of the calculus-based, enumerative and stochastic approaches.....	31
4.2	Efficiency of the calculus-based, enumerative and stochastic approaches.....	31

---

## ***Chapter 1 – Introduction***

The simultaneous localization and mapping (SLAM) problem is one of the major research areas in the robotic community, which is a key requirement for the development of truly autonomous robots. The SLAM is a technique used by robots to 1) continuously locate itself in an unknown environment using its sensor inputs and 2) build its own map at the same time for future localization or navigation. The navigation environment is described by a set of landmarks, while the nature of landmark depends on the sensor type. When the robot explores a new place, new landmarks are added to the map and the map is built incrementally. Based on the estimated location and the map built, the robot can navigate around the environment and execute various commands autonomously without any prior information on the environment. SLAM is a complicated task that can be divided into three major steps: 1) knowledge acquisition, 2) robot localization and 3) map building and management. These steps are applied repeatedly during the SLAM process.

In knowledge acquisition, information (landmarks) is extracted from the input sensor data. The representations of landmarks depend on the sensor type and localization algorithm. For example, landmarks for image are commonly represented as corner, line, pattern and the recently proposed scale-invariant feature [Lowe, 2004]. In the case of laser range sensor, landmarks can be represented by the raw range scan or the foreground points and edges which are extracted by range data clustering [Bailey *et al.*, 1999], etc.

In map building and management, the map is built continuously by adding new landmarks that the robot explored during its navigation. A new landmark is located using the current map and robot state before adding to the current map. Sometimes a new landmark needs to be observed for a period of time before it can be located in order to increase its certainty. As the number of landmarks increases along the navigation, the computational cost and map storage increases exponentially so that effective map management algorithms are necessary. [Guivant and Nebot, 2001] proposed to divide the global map into many local regions so that the computational cost of SLAM can be reduced. [Dissanayake *et al.*, 2000] suggested removing landmarks according to their information content to minimize the loss of information.

---

Robot localization is to estimate the robot state (e.g. position, orientation, velocity, acceleration, etc.) continuously according to the extracted landmarks, the current map and the previous robot states. The localization process includes 1) local localization and 2) global localization. Local localization is to keep track of the robot locally when the robot is under normal navigation. Global localization is to localize the robot within the whole map when the robot is first put into the map or its track is lost during navigation. Both of them are important in robot localization as global localization allows the robot to be re-localized when it is lost, while local localization allows the robot to be localized efficiently most of the time.

The SLAM technique has a wide range of applications. It has been applied to 1) autonomous service-robots used in public areas, such as museums [Graf *et al.*, 2004] [Dellaert *et al.*, 1999] and home stores [Gross *et al.*, 2001], 2) wearable visual robotics [Davison *et al.*, 2003] which estimates the motion of the wearer and detects the environment. It is useful in remote collaboration and augmented reality and 3) autonomous vehicles that can navigate automatically [Dissanayake *et al.*, 2001][Chou *et al.*, 2004]. These applications show the demands of high quality SLAM techniques.

Recently, [Liu *et al.*, 2006] developed a telemedicine system for remote health and activity monitoring that targets the elderly and patients at home. The system allows medical professionals to deliver health care and to share medical information remotely by making use of the well-developed telecommunication technologies. On the client side, wireless sensors are carried by the patient for biological data collection. On the remote side, medical professionals monitor patients' situations by means of multiple vital-sign parameters (i.e. electrocardiography, blood pressure, heart rate, etc.) via laptop, desk PC, mobile phone or PDA. Moreover, a robot pet is incorporated in the client side of the system for the following functions: First, the robot pet can behave autonomously and acts as a companion for the patient. Second, it can be remotely controlled (taking pictures, walking and speaking) via computer or mobile devices, which serves as a patient monitoring unit as well as a mode of communication between patients and healthcare providers. Lastly, the robot pet collects biological data from wireless sensors and transmits it to the remote healthcare provider via its built-in WLAN card.

---

In order to be autonomous and to cope with the aforementioned tasks, the robot pet should be able to localize itself in the environment. Ability of the robot to localize itself with no prior information is even better, so that the robot pet can work properly wherever it is putting in. Wheeled robot localization has been discussed for years. Recently, many researchers have shifted their focus to legged robots as it is more suitable to operate in a practical environment which involves uneven terrain and stairs, while wheeled robots need to be navigated in relatively even surfaces. Furthermore, human beings are more preferable to work or interact with animal- or human-like legged robots than wheeled robots. Besides the appearance, legged robots are also desired to behave autonomously. To be autonomous, the ability to localize itself is an essential requirement. However, localization of the legged robot is more difficult than that of the wheeled robot in the following aspects: First, walking motion of legged robots usually involve leg slippages which lead to unpredicted and unmodeled movements. These result in robot position and orientation different from those expected. If no proper localization method is applied, the robot position error will accumulate and increase along the track. Second, due to the oscillated walking motion of legged robots, sensors are vibrated so that the collected data is fluctuated. In addition, as sensors move in a 3D space with six degrees of freedom (DOFs), the complexity of the localization problem is increased.

Information from the surroundings for robot localization is received by sensors that are equipped on the robot. Sensors such as laser systems, cameras and sonar sensors are commonly used for robot localization. The accuracy of a localization algorithm is highly dependent on the quality of sensor data. However, quantity and size of sensors equipped on a robot are limited by the robot size. Large sensors, such as laser system, are not feasible for small home-use robots like the one used in [Liu *et al.*, 2006]. Though images contain massive amounts of information (corner, color, pattern, intensity and etc.) that is useful in localization, there are still many difficulties in visual localization. First, images can only be used for localization after the landmarks (e.g. corner and pattern) are extracted by some feature extraction techniques. However, the extraction result is highly affected by lighting condition and image noise. Incorrect landmark detections affect the result of localization and mapping directly. Second, compare with laser system which can receive data from

---

360 degrees, the field of view of a camera is small. The amount of information received in an instant is limited. Lastly, image points from single image are lack of depth information. As our robot pet is equipped with a single low-resolution camera as the only sensor, we are interested in the visual localization method for small walking robot using single camera.

In this thesis, we present a vision-based localization algorithm for a small walking robot by assuming that the robot navigation environment consists of a set of landmarks with known positions. We propose to formulate the vision-based localization for a high-dimensional movement robot as an optimization. Afterward, the formulated objective function is optimized using a genetic algorithm (GA). Given the feature coordinates of an image captured by the equipped camera at current instant and the corresponding landmark positions, the localization algorithm is able to estimate the current robot state (i.e. position and orientation).

The thesis is organized as follows: In chapter 2, a brief review of the recent proposed localization algorithms is presented. These algorithms include the extended Kalman filter (EKF)-based localization algorithm and the Monte Carlo Localization algorithm. In chapter 3, the vision-based localization problem for legged robot is defined. We show that the vision-based localization for a high DOFs moving robot can be formulated as an optimization. Thus, the robot state can be obtained by optimizing the formulated objective function. In chapter 4, mechanism of some commonly used search algorithms are presented. Their limitations and efficiency are discussed. Afterward, the possible choice of search algorithms for the objective function proposed in chapter 3 is analyzed. In chapter 5, the mechanism of an evolutionary based optimization technique, genetic algorithm, is presented. In the proposed algorithm, a genetic algorithm is employed to solve the objective function. The details of the employment including chromosome formation, fitness functions, genetic operators and selection scheme are discussed. Based on the basic principle of the genetic algorithms, we propose several approaches to increase the efficiency of the localization algorithm. In addition, a search space defining method, called adaptive search space strategy is presented, which aims to increase the efficiency of the proposed algorithm as well as weaken the influence of noisy feature points. In chapter 6, the performance of the proposed algorithm is measured in terms of



accuracy, efficiency, noise sensitivity and noise reduction ability by three experiments. In the first experiment, the accuracy and efficiency of the proposed algorithm is tested by a simulation of robot localization in an area of  $10\text{m} \times 10\text{m}$  for 60 seconds. In the second experiment, the noise sensitivity of the proposed algorithm is examined by simulations similar to that of the first experiment except that different levels of Gaussian noise are added to the input feature points. In the third experiment, the noise reduction performance of the proposed adaptive search space strategy is illustrated by simulation of robot state estimation using localization algorithm that is not applied with the adaptive search space strategy. Finally, the major contributions of the thesis are concluded in chapter 7.

## Chapter 2 – State of the art in Vision-based Localization

Many localization algorithms for robots equipped with single camera have been proposed. Since legged robot navigation is a fairly new field, most of the localization approaches are developed for wheeled robot. They can be classified into two groups: 1) Extended Kalman Filter-based localization and 2) Monte Carlo Localization. In the rest of this chapter, overviews of these recently proposed localization algorithms are presented.

### 2.1 Extended Kalman Filter-based Localization

The Kalman filter (KF) is first introduced by [Kalman 1960], which is a recursive filter that estimates the state of a linear dynamical system with noisy measurements. The Extended Kalman filter (EKF) is an enhanced version of the Kalman filter, which can deal with non-linear dynamical system. The EKF has been the subject of extensive research in the area of autonomous robotics. An overview of the EKF and the process of the EKF-based localization are given in the rest of this section.

#### 2.1.1 Overview of the EKF algorithm

The EKF targets the problem of trying to estimate the “state” of a discrete-time controlled process that is governed by a stochastic difference equation [Welch and Bishop 2004]. Suppose the state is  $\xi \in \mathfrak{R}^m$  and it is governed by the stochastic difference equation:

$$\xi_k = f(\xi_{k-1}, \mathbf{u}_{k-1}, \mathbf{w}_{k-1}) \quad (2.1)$$

with a measurement  $\mathbf{z} \in \mathfrak{R}^n$  that is:

$$\mathbf{z}_k = h(\xi_k + \mathbf{v}_k) \quad (2.2)$$

, where  $\mathbf{w}_k \in \mathfrak{R}^m$  and  $\mathbf{v}_k \in \mathfrak{R}^n$  are zero-mean random variables that represent the process noise and the measurement noise respectively,  $\mathbf{u}_k \in \mathfrak{R}^p$  is the control input and the subscript  $k$  indicates the time step. Then, the process noise covariance matrix  $\mathbf{R}_k$  and the measurement noise covariance matrix  $\mathbf{Q}_k$  are  $m \times m$  matrix and  $n \times n$  matrix respectively which might be fixed or varied with time. The non-linear difference function  $f$  relates the state at the previous time step  $k-1$  to the state at the current time step  $k$ . Meanwhile, the non-linear function  $h$  related the state and the measurement. Though the noise  $\mathbf{w}_k$  and  $\mathbf{v}_k$  are unknown, the prior state  $\tilde{\xi}_k^-$  and the measurement at time step  $k$  can be estimated using knowledge prior to time step  $k$  by setting  $\mathbf{w}_k$  and  $\mathbf{v}_k$  to zero:

$$\tilde{\xi}_k^- = f(\tilde{\xi}_{k-1}, \mathbf{u}_{k-1}, 0) \quad (2.3)$$

$$\tilde{\mathbf{z}}_k = h(\tilde{\xi}_k^-, 0) \quad (2.4)$$

, where  $\tilde{\xi}_k$  is the posterior state at time step  $k$ . Then, suppose the actual state at time step  $k$  is  $\xi_k$ , the prior estimate error  $\mathbf{e}_k^-$  and the posterior estimate error  $\mathbf{e}_k$  are defined as:

$$\mathbf{e}_k^- = \xi_k - \tilde{\xi}_k^- \quad (2.5)$$

$$\mathbf{e}_k = \xi_k - \tilde{\xi}_k. \quad (2.6)$$

Moreover, the prior error covariance matrix  $\mathbf{P}_k^-$  and the posterior estimate error covariance matrix  $\mathbf{P}_k$  are defined as:

$$\mathbf{P}_k^- = E[\mathbf{e}_k^- \cdot \mathbf{e}_k^{-T}] \quad (2.7)$$

$$\mathbf{P}_k = E[\mathbf{e}_k \cdot \mathbf{e}_k^T]. \quad (2.8)$$

Given the knowledge of measurement  $\mathbf{z}_k$ , the posterior state  $\tilde{\xi}_k$  is obtained by:

$$\tilde{\xi}_k = \tilde{\xi}_k^- + \mathbf{K}_k (\mathbf{z}_k - \tilde{\mathbf{z}}_k) \quad (2.9)$$

where the difference ( $\mathbf{z}_k - \tilde{\mathbf{z}}_k$ ) is the residual and  $\mathbf{K}_k$  ( $m \times n$  matrix) is the Kalman gain which is chosen that the posterior error covariance matrix  $\mathbf{P}_k$  is minimized. As  $\mathbf{K}_k$  is with respect to  $\mathbf{R}_k$ , it can be regarded as a weight of the residual which depends on the noise level of the measurement. A popular form of  $\mathbf{K}_k$  is:

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^T (\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{V}_k \mathbf{R}_k \mathbf{V}_k^T)^{-1} \quad (2.10)$$

, where  $\mathbf{H}_k$  ( $n \times m$  matrix) and  $\mathbf{V}_k$  ( $m \times n$  matrix) are the Jacobian matrices of partial derivatives of  $h$  with respect to  $\xi_k$  and  $\mathbf{v}_k$  respectively.

### 2.1.2 Process of the EKF-based localization algorithm

The EKF-based localization is a recursive solution to robot localization with noisy sensor measurements. The EKF aims to estimate the “state” of a system, where the nature of the state depends on the system. For robot localization, the state of the system is the robot state which can include the robot position, orientation, velocity or acceleration. The robot state at a time step is estimated in two phases: 1) predict phase and 2) update phase. The predict phase is to project forward the robot state and the error covariance matrix in the previous time step in order to obtain a prior robot state and error covariance matrix for the current time step. The update phase is to use the knowledge of the measurement in the current time step to correct the prior robot state and error covariance matrix in order to obtain an improved posterior robot state and error covariance matrix.

#### *Predict phase*

In the predict phase, the prior robot state  $\tilde{\xi}_k^-$  and error covariance matrix  $\mathbf{P}_k^-$  are estimated without using the current measurement. The  $\tilde{\xi}_k^-$  is estimated according to Eq. (2.3). Meanwhile, the  $\mathbf{P}_k^-$  is computed by:

$$\mathbf{P}_k^- = \mathbf{A}_k \mathbf{P}_{k-1} \mathbf{A}_k^T + \mathbf{W}_k \mathbf{Q}_{k-1} \mathbf{W}_k^T. \quad (2.11)$$

, where  $\mathbf{Q}_k$  is the process noise covariance matrix ,  $\mathbf{A}_k$  ( $m \times m$  matrix) is the Jacobian matrix of partial derivatives of  $f$  with respect to  $\xi_k$  and  $\mathbf{W}_k$  ( $m \times m$  matrix) is the Jacobian matrix of partial derivatives of  $f$  with respect to  $\mathbf{w}_k$ .

### *Update phase*

In the update phase, given the measurement  $\mathbf{z}_k$ , the posterior robot state  $\tilde{\xi}_k$  and error covariance matrix  $\mathbf{P}_k$  are obtained. The  $\tilde{\xi}_k$  is computed by:

$$\tilde{\xi}_k = \mathbf{K}_k (\mathbf{z}_k - h(\tilde{\xi}_k^-, 0)) \quad (2. 12)$$

, where the Kalman gain  $\mathbf{K}_k$  is obtained as described in Eq. (2. 10). Meanwhile, the  $\mathbf{P}_k$  is computed by:

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_k^- \quad (2. 13)$$

The EKF-based localization algorithm tracks the robot state by applying the predict and update phase at each time step. To summarize, the flow diagram of the EKF-based localization algorithm is shown in Fig. 2. 1.

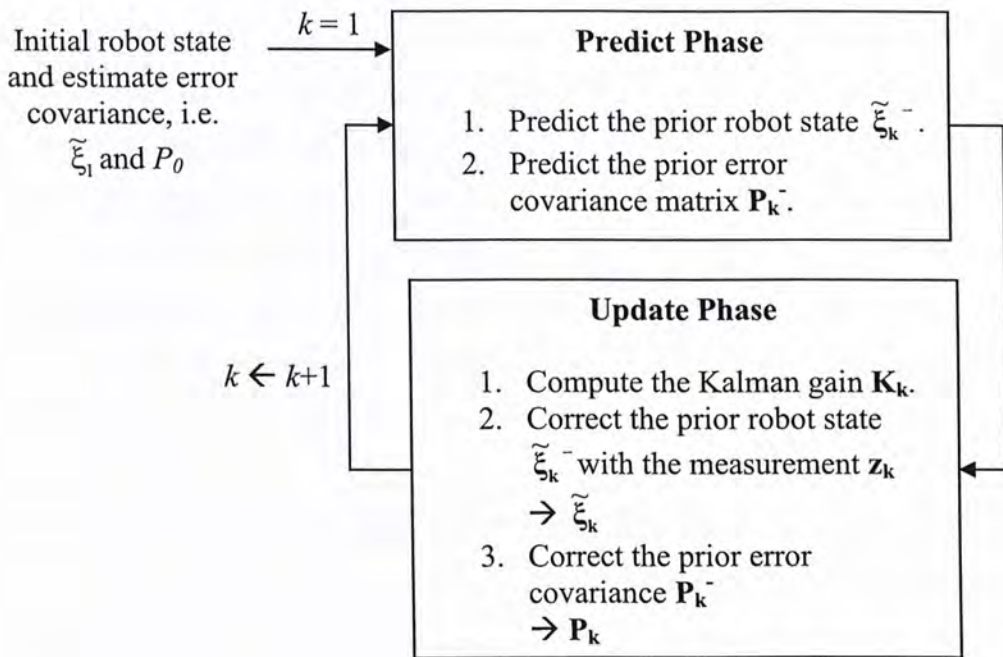


Fig. 2. 1 Flow diagram of the Extended Kalman filter – based localization algorithm.

### 2.1.3 Recent EKF-based vision-based localization algorithms

Recently, some visual localization algorithms using the EKF are proposed. [Karlsson *et al.*, 2005] proposed a SLAM algorithm based on visual and odometric signals. It enables robot navigation in populated and dynamic areas. [Bonci *et al.*, 2005] proposed a SLAM algorithm using sensor data from odometry, camera and sonar. The reliability of visual landmark position is improved by including the landmark positions in the state of the EKF. In these two examples, the robots were equipped with camera and other sensors, and the EKF was used as a tool to fuse data from different sensors. On the other hand, some researchers proposed that SLAM algorithm using camera as the only sensor is possible. For example, [Wang *et al.*, 2005] proposed a visual localization algorithm using the landmarks on a planar ground. Also, [Jeong and Lee, 2005] proposed a ceiling vision-based SLAM algorithm which localizes a robot using images captured by the equipped camera that is mounted in a direction facing the ceiling. One of the problems of single camera localization is the lack of depth information of image points. The approaches proposed in [Wang *et al.*, 2005] and [Jeong and Lee, 2005] solve the problem by assuming that the landmarks are always on the same plane (ceiling or ground). The

aforementioned algorithms were applied to wheeled robots, whose number of DOF is less than that of legged robots. Besides the applications on wheeled robot localization, an EKF-based localization approach has been proposed and applied to a six-legged walking robot LAURON III [Gassmann *et al.*, 2005] with the fusion of GPS and odometry measurement. Though EKF-based visual localization approach for the walking robot is not found, a real-time visual SLAM approach for a smoothly free-flying handheld camera is proposed by [Davison, 2003]. Similar to the camera equipped on walking robot, the handheld camera involves high-dimensional movement.

#### **2.1.4 Advantages of the EKF-based localization algorithm**

The EKF has been proven an effective solution to real-time robot localization in many researches. As the EKF-based localization algorithm computes the robot state at a time step by the measurement at that time step only, the computational cost is relative small. Moreover, the EKF-based algorithm does not have the assumption of rigid environment (landmarks). As the EKF-based algorithm aims to estimate the “state” of the system at each time step, things in the system that with interests can be added to the state. In the case of dynamic environment, the landmark positions can be included in the state, which will be updated at each time step. Furthermore, the EKF-based algorithm is a tool to fuse sensor measurements when more than one types of sensor are used for localization.

#### **2.1.5 Disadvantages of the EKF-based localization algorithm**

The EKF-based localization algorithm use unimodal Gaussian to represent the distribution of the state while the actual state distribution might be non-Gaussian. Thus, the EKF-based algorithm is always used for robot tracking but not global localization. Moreover, the EKF-based algorithm assumes smooth robot movement as the predict phase estimates the robot state at current time step by the robot state at the previous time step. To ensure that the robot states at successive time steps are continuous, the robot movement should be smooth. Therefore, the EKF-based algorithm is usually applied to wheeled robot instead of legged robot. As legged robot as well as its sensors vibrates during navigation, robot velocities at successive

frames are not continuous. In this case, robot state estimated in the predict phase might involve large error. If the error cannot be corrected during the update phase due to noisy measurements, the localization error will be accumulated along the track. Since EKF-based localization does not deal with the global localization problem, the robot track might be lost when more errors have accumulated.

## 2.2 Monte Carlo Localization

Monte Carlo Localization (MCL) is a robot self-localization approach first introduced by [Fox *et al.* 1999], which is a recursive filter that estimates the robot state given the sensor measurements. It is a sampling-based approach that the probability density function (p.d.f.) of the robot is approximated by a set of particles. MCL can cope with both of the global and local localization problem. An overview of the MCL and the recent works related to the MCL are discussed below.

### 2.2.1 Overview of MCL

In the MCL, given the sensor measurements collected from the starting time to the time step  $k$ , the posterior p.d.f. of the robot state at time step  $k$  can be represented by a set of random and weighted particles  $\mathbf{S}_k = \{s_{i,k}\}_{i \in [1, N_c]}$ , where the subscript  $k$  indicates the time step and  $N_c$  is the number of particles. Each particle  $s_{i,k}$  has a weighting factor  $p_i$ , where  $p_i \geq 0$  and  $\sum_{i=1}^{N_c} p_i = 1$ . Suppose the position of a sample  $s_i$  is equal to  $\mathbf{l}_i$ ,  $\mathbf{l}_i$  denotes the robot state and is in the form:  $\mathbf{l}_i = [x, y, \theta]$  (i.e.  $[x, y]$  is the robot position vector and  $\theta$  is the robot orientation).

The MCL algorithm localizes the robot by recursively computing the p.d.f. of the robot state at each time step. The determination of a robot state p.d.f. can be divided into two phases: 1) Robot motion and 2) Sensor readings.

#### *Robot motion*

In the robot motion phase, a prior particle set  $\mathbf{S}_k^- = \{s_{i,k}^-\}_{i \in [1, N_c]}$  that estimates the p.d.f. of the robot state at time step  $k$  using the input motion command and the



sensor data prior to time step  $k$ , is generated when the robot moves. Particles of  $\mathbf{S}_k^-$  are generated by selecting particles from  $\mathbf{S}_{k-1}$  and applying the input control command to each particle, where  $\mathbf{S}_k$  is the posterior particle set at time step  $k$ . In probabilistic representation, for each particle in  $\mathbf{S}_{k-1}$ , a particle of  $\mathbf{S}_k^-$  is generated by selecting a particle from the density  $p(\mathbf{L}_k | \mathbf{a}_{k-1}, s_{i,k-1})$ , where  $\mathbf{L}_k$  is the robot state at time step  $k$  and  $\mathbf{a}_{k-1}$  is the input control command. The newly generated particles in  $\mathbf{S}_k^-$  are equally weighted, i.e.  $p_i = 1/N_c, i \in [1, N_c]$ .

### **Sensor reading**

In the sensor reading phase, the posteriori particles set  $\mathbf{S}_k$  that describes the p.d.f. of the robot state is determined by considering the sensor measurement at time step  $k$ , namely  $\mathbf{b}_k$ . First, each particle  $s_{i,k}$  in  $\mathbf{S}_k^-$  are re-weighted by the likelihood that the robot state is at its particle position  $\mathbf{l}_{i,k}$  given the sensor measurement  $\mathbf{b}_k$ , i.e.  $p_i \leftarrow \tau_i p(\mathbf{b}_k | \mathbf{l}_{i,k}), i \in [1, N_c]$ , where  $\tau_i$  is the normalization factor that is to ensure that  $\sum_{i=1}^{N_c} p_i = 1$  after the re-weighting process. The weighted particles in  $\mathbf{S}_k^-$  reflect the robot state p.d.f. change with the knowledge gained from  $\mathbf{b}_k$ . Then, the re-sampling process randomly selects particles from the weighted  $\mathbf{S}_k^-$  with the likelihood depends on their  $p$ -values, i.e. particle with larger  $p$ -value is more probably to be selected. The resultant particle set formed by the selected particles is the posterior particle set  $\mathbf{S}_k$  for time step  $k$ . The weighting factors of the particles in  $\mathbf{S}_k$  are reset to be equal, i.e.  $p_i = 1/N_c, i \in [1, N_c]$ .

To summarize, the flow diagram of the MCL algorithm is shown in Fig. 2. 2. In the case that the initial state of the robot is unknown, the process starts with a set of particles that are randomly generated in the navigation environment.

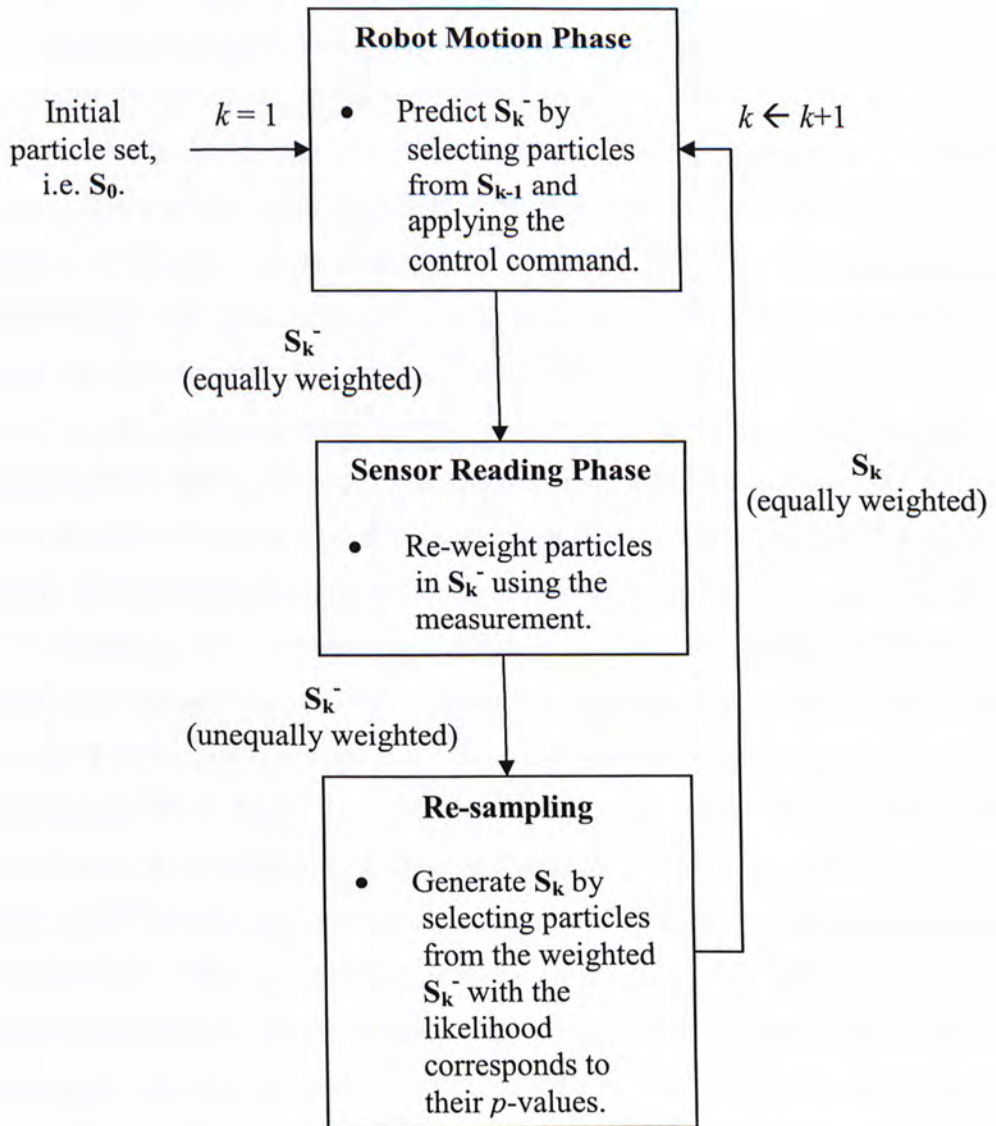


Fig. 2. 2 Flow diagram of the Monte Carlo Localization algorithm.

### 2.2.2 Recent MCL-based localization algorithms

The MCL was first applied to Minerva, a wheeled robot which was employed as tour-guide in a museum [Fox *et al.*, 1999]. The robot was equipped with a camera pointed toward the ceiling. Using the ceiling images and the motions recorded by its odometry, the algorithm is able to track the robot path even though the motions recorded by the robot's odometry involved significant errors. Furthermore, [Fox *et al.*, 1999] showed that the MCL algorithm is able to globally localize a robot. They proposed that the size of the particle set should be varied from tracking to global localization.

Besides localization for wheeled robot, MCL algorithm is also applied to legged robot localization. Several researchers: [Rofer and Jungle, 2003], [Lenser and Veloso, 2000], [Ueda *et al.*, 2003] and [Sridharan *et al.*, 2005] used the MCL as a baseline for robot localization in the RoboCup Sony Legged Robot League [RoboCup]. It is a soccer game participated by teams of autonomous legged robots, which requires the robots to localize itself in real-time. The RoboCup environments are approximately 4m x 6m with color-coded landmarks. The robots participating in the game are four-legged and equipped with single camera in their heads. The equipped camera captures images as sensor data for localization during the game. [Rofer and Jungle, 2003] proposed that the stability of the MCL algorithm can be increased by limiting the change of the weighting factor of each particle to a certain maximum. This weakens the effect of measurement errors on the particle weights, while maintaining the re-localizing efficiency after kidnapping. Kidnapping describes the situation that a robot is picked up and placed at a different position. [Lenser and Veloso, 2000] proposed the Sensor Resetting Localization (SRL) which is an extension of the MCL. The SRL handles the unmodelled movements using fewer particles by replacing the small  $p$ -valued particles with the particles selected from the p.d.f. given by the sensor measurement during the re-sampling process. [Sridharan *et al.*, 2005] proposed several enhancements to the MCL which aim to improve the localization accuracy. One of the enhancements is “Landmark histories” which suggests that by storing information of the observed landmarks in successive frames, it is possible to localize the robot when inadequate landmarks are seen simultaneous. The aforementioned examples demonstrate that the MCL-based visual localization algorithms are sufficient for real-time robot localization in the RoboCup environment.

### 2.2.3 Advantages of the MCL-based algorithm

Similar to the EKF-based algorithm, the MCL is an efficient localization algorithm which can localize the robot in real-time. Since MCL represents the p.d.f. of robot state by a set of particles, the robot state p.d.f. can be described by multi-modal distribution. Hence, the MCL is able to cope with the global localization

problem. That is, the robot can be localized without knowing the initial position or be re-localized when the robot track is lost.

#### 2.2.4 Disadvantages of the MCL-based algorithm

The accuracy of the MCL-based algorithm is highly dependent on the density of the particle. The estimated robot state is accurate only if the blob of particles at the actual robot state is so dense that it can be distinguished from other particles. Several aforementioned examples illustrate that the visual localization algorithms based on the MCL performs well on legged robots. However, the Robocup environment is relatively small. In contrary to the environment of RoboCup, practical environments are usually larger and the landmarks are further from the robot that the expected observation is less sensitive to the particle position. In other words, particles at positions around the actual robot state have similar  $p(\mathbf{b}_k | \mathbf{l}_{i,k})$  values, i.e.  $p$ -values. Hence, the particles at the actual robot state are not distinct from the other particles around the actual robot state during the re-weighting and re-sampling process. As a result, a loosely grouped blob of particles near the actual robot state is generated after the re-sampling. As the robot state is estimated by the density of the resultant particles, a loosely grouped blob of particles reduces the accuracy of the estimated robot state.

Another disadvantage of the MCL-based algorithm is that the number of particles needed is exponentially increased with the number of dimensions of the state. (i.e. the number of particle is  $n^m$  for  $m$ -dimension state, where  $n$  is the number of particles for one-dimension state. )

### 2.3 Summary

The commonly used visual robot localization algorithms are classified into two types: 1) EKF-based localization and 2) MCL. The EKF-based localization algorithm assumes smooth robot motion. Hence, it is not suitable for wheeled robot localization. Moreover, the EKF-based localization algorithm does not cope with global localization problem. The MCL algorithm can cope with both global and local localization, but it is only suitable for relatively small navigation environment.

Seen from the drawbacks of the previous works, an ideal visual localization algorithm for legged robots should: 1) be accurate and fast, 2) localize the robot using natural landmarks in the environment regardless of their distances from the robot, 3) adapt to the oscillating sensor data due to rapid changes of the robot velocity and 4) re-localize the robot state quickly after the robot is kidnapped.

---

## ***Chapter 3 – Vision-based Localization as an Optimization Problem***

The vision-based localization is to estimate the states (positions and orientations) of the robot given the landmark positions and the feature points extracted from the captured images. In this problem, the state (position and orientation) of the equipped camera is first estimated, and the state of the robot body can be obtained by the transformation of the relative position and orientation between the camera and the robot body. For legged robot, the relative position between the camera and the robot body can be varied from time to time. Compare between legged robot and wheel robot localizations, the degree of freedom (DOF) of the legged robot is usually higher than that of wheeled robot. Hence, the dimensionality of the robot (camera) state in legged robot localization is usually higher. The robot (camera) state in wheeled robot localization is usually 3D, i.e. 2D of position and 1D of orientation. Though the orientation of the camera equipped on the legged robot might depend on the statuses of several movable joints of the robot, the localization problem is simplified without loss of generality by using a vector that represents orientation in 3D space to represent the camera orientation. Meanwhile, the camera position is represented by a 3 by 1 vector. In this chapter, we show that the vision-based localization problem for a high DOFs moving robot can be formulated as an optimization. By solving the objective function, the camera state can be obtained.

### **3.1 Relationship between the World, Camera and Robot Body Coordinate Systems**

To model the vision-based localization as an optimization, three coordinate systems: the world coordinate system  $W$ , camera coordinate system  $C$  and the robot body coordinate system  $B$  are defined. These coordinate systems are all represented in the Cartesian coordinate system.

The world coordinate system is defined based on the robot navigation environment. The  $xz$ -plane of  $W$  is parallel to the floor of the navigation environment (assuming that the floor of the navigation environment is flat). The objective of the localization is to obtain the states (i.e. position and orientation) of the robot (i.e. center of the robot body) related to  $W$ . The “center of the robot body” and “state of the center of the robot body” are abbreviated as “robot” and “robot state” respectively in the following discussion. By formulating the vision-based localization problem as an optimization, and optimizing the objective function, the state (position and orientation) of the equipped camera related to  $W$  can be obtained. The robot state can be computed by transformation between the camera and the robot if the relative position and orientation between the equipped camera and the robot is known.

To illustrate the transformation between the camera and the robot, the camera coordinate system  $C$  and the robot body coordinate system  $B$  are defined. The origin of  $C$  is the camera position (see Fig. 3. 1 ).

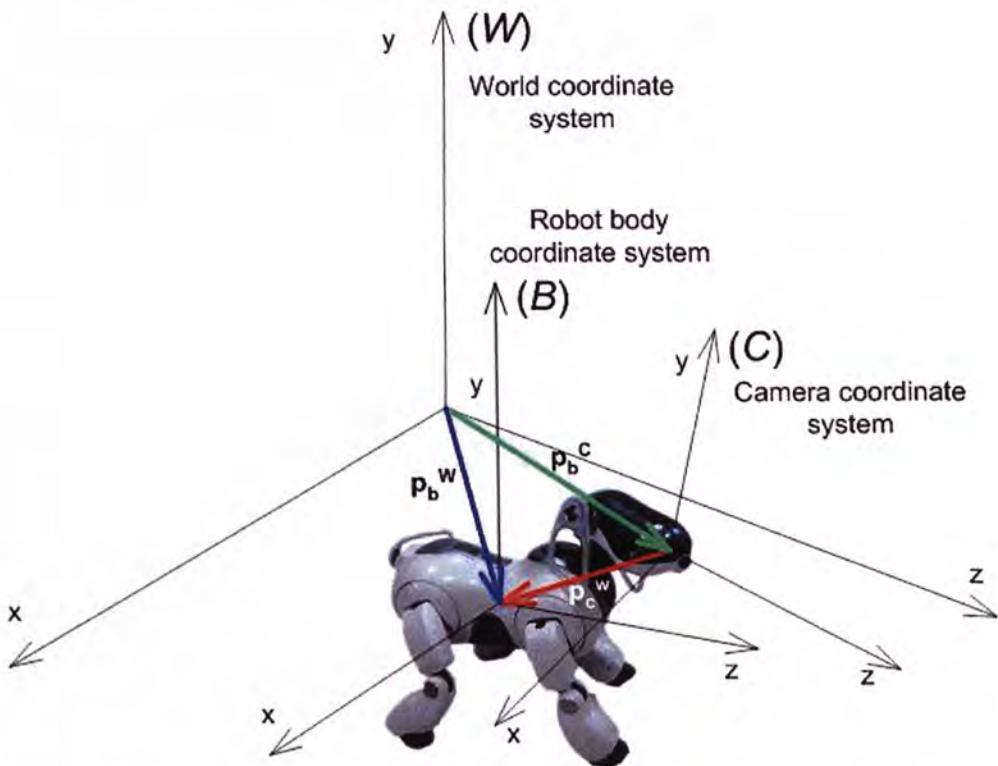


Fig. 3. 1 Relationship between the world coordinate system  $W$ , the camera coordinate system  $C$  and the robot body coordinate system  $B$ .

Suppose the camera position vector related to  $W$  is  $\mathbf{p}_c^W$  (the superscript indicates the coordinate system that the vector is related to) and the rotation matrix from  $W$  to  $C$  is  $\mathbf{R}_{wc}$  (a  $3 \times 3$  matrix), the relation between  $W$  and  $C$  can be described by the following equation:

$$\mathbf{S}^C = \mathbf{R}_{wc} (\mathbf{S}^W - \mathbf{p}_c^W) \quad (3.1)$$

, where  $\mathbf{S}^C$  and  $\mathbf{S}^W$  are the 3D point at  $C$  and  $W$  respectively. Hence, if  $\mathbf{S}^W$  is the camera position (i.e.  $\mathbf{S}^W = \mathbf{p}_c^W$ ),  $\mathbf{S}^C$  is equal to  $[0, 0, 0]$  (i.e. origin of  $C$ ). The origin of  $B$  is the center of the robot body (see Fig. 3. 1 ). Suppose the robot position vector related to  $C$  is  $\mathbf{p}_b^C$  and the rotation matrix from  $C$  to  $B$  is  $\mathbf{R}_{cb}$  ( $3 \times 3$  matrix), the relationship between  $B$  and  $C$  can be described by the following equation:

$$\mathbf{S}^B = \mathbf{R}_{cb} (\mathbf{S}^C - \mathbf{p}_b^C) \quad (3.2)$$

, where  $\mathbf{S}^B$  and  $\mathbf{S}^C$  are the 3D point at  $B$  and  $C$  respectively.

We define the robot position vector related to  $W$  as  $\mathbf{p}_b^W$  and the robot orientation represented by rotation matrix from  $W$  to  $B$  as  $\mathbf{R}_{wb}$  ( $3 \times 3$  matrix), such that

$$\mathbf{S}^B = \mathbf{R}_{wb} (\mathbf{S}^W - \mathbf{p}_b^W). \quad (3.3)$$

Suppose  $\mathbf{p}_c^W$  and  $\mathbf{R}_{wc}$  are known by optimizing the objective function, and  $\mathbf{p}_b^C$  and  $\mathbf{R}_{bc}$  are obtained from the statuses of the controllable DOFs of the robot,  $\mathbf{p}_b^W$  and  $\mathbf{R}_{wb}$  can be computed as follows:

First, substitute Eq. (3. 1) to Eq. (3. 2),

$$\mathbf{S}^B = \mathbf{R}_{cb} (\mathbf{R}_{wc} (\mathbf{S}^W - \mathbf{p}_c^W) - \mathbf{p}_b^C).$$

Afterward, re-arrange the coefficients as:



$$\begin{aligned}
\mathbf{S}^B &= \mathbf{R}_{cb} \bullet \mathbf{R}_{wc} ( \mathbf{S}^W - \mathbf{p}_c^W ) - \mathbf{R}_{wc}^{-1} \bullet \mathbf{p}_b^C \\
&= \mathbf{R}_{cb} \bullet \mathbf{R}_{wc} ( \mathbf{S}^W - ( \mathbf{p}_c^W + \mathbf{R}_{wc}^{-1} \bullet \mathbf{p}_b^C ) ).
\end{aligned} \tag{3.4}$$

By comparing the coefficients of Eq. (3.4) and Eq. (3.3),

$$\mathbf{R}_{wb} = \mathbf{R}_{cb} \bullet \mathbf{R}_{wc} \tag{3.5}$$

$$\mathbf{p}_b^W = \mathbf{p}_c^W + \mathbf{R}_{wc}^{-1} \bullet \mathbf{p}_b^C. \tag{3.6}$$

### 3.2 Formulation of the Vision-based Localization as an Optimization Problem

As discussed in the previous section, the localization of a robot is equivalent to the localization of the equipped camera. In this section, we focus on the estimation of the state (position and orientation) of the equipped camera at a certain time step using the image captured by the camera at that same time step. It is assumed that there is a set of landmarks with known positions in the robot navigation environment. The feature points corresponding to these landmarks are extracted from the captured image and used as the sensor measurements for the localization. It is also assumed that the captured image always contains the feature points of some of the landmarks.

Due to the oscillated walking motion of the legged robot, the camera vibrates so that its position and orientation change quickly. Therefore, the translation of the camera is no longer on a planar surface. Instead, the camera involves three degrees of translation. Hence, the camera position vector  $\mathbf{p} \in \mathfrak{R}^3$  related to  $W$  is defined as:

$$\mathbf{p} = [p_x \ p_y \ p_z]^T \tag{3.7}$$

, where  $p_x$ ,  $p_y$  and  $p_z$  are the coordinates along the  $x$ -,  $y$ -, and  $z$ - axis of  $W$  respectively. Similarly, the degree of camera rotation increase due to the oscillated walking motion of the legged robot. Hence, the camera has three degree of rotation. The camera orientation vector  $\mathbf{o} \in \mathfrak{R}^3$  is represented by Euler angles as:

$$\mathbf{o} = [\theta_x, \theta_y, \theta_z]^T \quad (3.8)$$

, where  $\theta_x, \theta_y$  and  $\theta_z$  are the rotation about the  $x$ -,  $y$ - and  $z$ -axis of  $W$  respectively. The camera state vector  $\mathbf{r}$  is composed of the camera position vector and the camera orientation vector as:

$$\mathbf{r} = \begin{bmatrix} \mathbf{p} \\ \mathbf{o} \end{bmatrix} = [p_x \quad p_y \quad p_z \quad \theta_x \quad \theta_y \quad \theta_z]^T \quad (3.9)$$

The camera position is  $\mathbf{p}$  related to  $W$  and it becomes the origin in  $C$ . According to Eq. (3. 1), the relation between  $W$  and  $C$  can be described by the following equation:  $\mathbf{S}^W = \mathbf{R} (\mathbf{S}^C - \mathbf{p})$ , where  $\mathbf{R}$  is the  $3 \times 3$  rotation matrix from  $W$  to  $C$  defined by  $\mathbf{o}$  as:

(By the right hand rule)

$$\mathbf{R} = \mathbf{R}_x \bullet \mathbf{R}_y \bullet \mathbf{R}_z \quad (3.10)$$

, where  $\mathbf{R}_x$ ,  $\mathbf{R}_y$  and  $\mathbf{R}_z$  are:

$$\mathbf{R}_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta_x) & \sin(\theta_x) \\ 0 & -\sin(\theta_x) & \cos(\theta_x) \end{bmatrix},$$

$$\mathbf{R}_y = \begin{bmatrix} \cos(\theta_y) & 0 & -\sin(\theta_y) \\ 0 & 1 & 0 \\ \sin(\theta_y) & 0 & \cos(\theta_y) \end{bmatrix},$$

$$\mathbf{R}_z = \begin{bmatrix} \cos(\theta_z) & \sin(\theta_z) & 0 \\ -\sin(\theta_z) & \cos(\theta_z) & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Suppose there is a set of landmark positions  $\mathbf{FN} = \{\mathbf{f}_i\}_{i \in [1, N_a]}$  where  $\mathbf{f}_i = [f_x^i, f_y^i, f_z^i]^T$  is the position vector of the  $i^{\text{th}}$  landmark related to  $W$  and  $N_a$  is the total number

of landmarks. The position vector of the  $i^{th}$  landmark  $\mathbf{f}_i$  related to  $C$  is defined as:  $\mathbf{h}_i = [h_x^i, h_y^i, h_z^i]^T$  (see Fig. 3. 2), i.e.  $\mathbf{h}_i = \mathbf{R}(\mathbf{f}_i - \mathbf{p})$ .

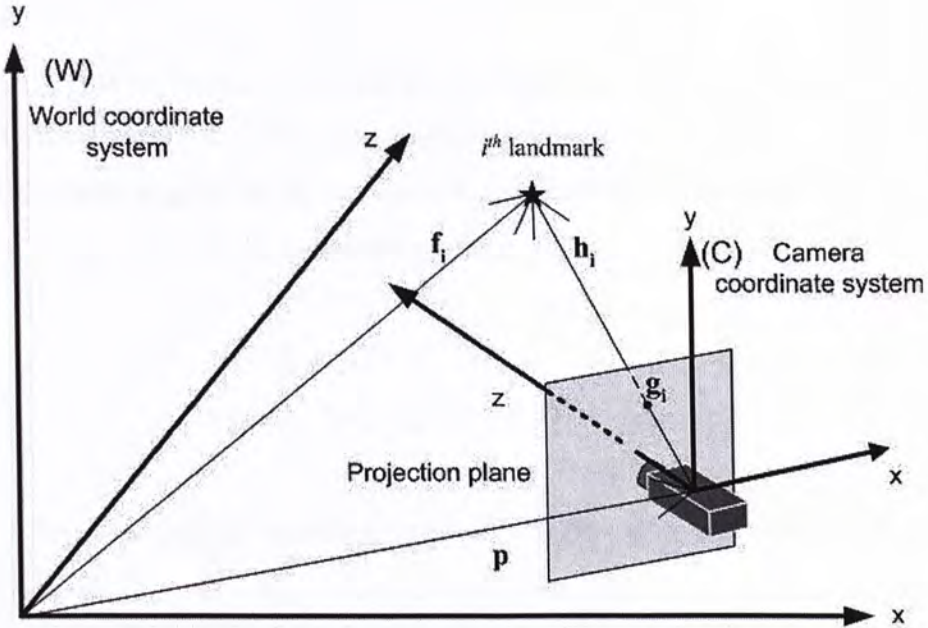


Fig. 3. 2 Relationship between the landmark position vector ( $\mathbf{f}_i$  and  $\mathbf{h}_i$ ) and the projected image point  $\mathbf{g}_i$ .

According to the pinhole camera model, the projection plane of the camera is perpendicular to the optical axis ( $z$ -axis of  $C$ ) and  $F$  cm apart from the center of projection (origin of  $C$ ), i.e. projection plane is  $z = F$ , where  $F$  is the focal length of the camera. The projected image point of  $\mathbf{h}_i$  on the projection plane related to  $C$  is defined as  $\mathbf{g}_i = [U_i, V_i, F]^T$  as shown in Fig. 3. 2, and  $[U_i, V_i]$  is related to  $\mathbf{h}_i$  by the following equation:

$$\begin{aligned} U_i &= F \frac{h_x^i}{h_z^i} \\ V_i &= F \frac{h_y^i}{h_z^i} \end{aligned} \quad (3. 11)$$

Suppose the feature coordinates correspond to the  $i^{th}$  landmark is  $\mathbf{m}_i = [u_i, v_i]^T$ ,  $\mathbf{m}_i$  is related to  $[U_i, V_i]$  by the following equation:

$$\begin{bmatrix} u_i \\ v_i \end{bmatrix} = \begin{bmatrix} c_x + \frac{U_i}{p_u} \\ c_y + \frac{V_i}{p_v} \end{bmatrix} \quad (3.12)$$

where  $[c_x, c_y]$  is the image coordinates of the principal point,  $p_u$  and  $p_v$  are the pixel length (cm per pixel) in  $x$ - and  $y$ - direction respectively.

By considering  $\mathbf{h}_i$  and its corresponding measured feature coordinates, namely  $\mathbf{m}_i' = [u_i', v_i']$ , we define the measured  $\mathbf{g}_i$ , namely  $\mathbf{g}_i'$  as:

$$\begin{aligned} \mathbf{g}_i' &= [U_i', V_i', F]^T \\ &= [p_u(u_i' - c_x), p_v(v_i' - c_y), F]^T \end{aligned} \quad (3.13)$$

A line equation that passes through  $\mathbf{h}_i$  and  $(\mathbf{h}_i - \mathbf{g}_i')$ , namely camera position line is formed, i.e.  $l_i: \mathbf{h}_i + t \cdot \mathbf{g}_i'$ , which describes the position of the camera. Since the measured feature coordinates  $\mathbf{m}_i'$  is assumed to be noiseless,  $\mathbf{g}_i$  is equal to  $\mathbf{g}_i'$  and  $L_i$  should pass through the origin of  $C$ . Suppose  $N_m$  ( $N_m \leq N_a$ ) feature points that correspond to the landmarks in **FN** are selected for localization, a set of  $N_m$  line equations  $\{L_i\}, i \in [1, N_m] \in \{l_j\}, j \in [1, N_a]$ , and  $L_j \neq L_k$  for  $j \neq k$  is formulated and they are converged at the origin, i.e. actual camera position. However,  $\mathbf{m}_i'$  always involves noise that  $\mathbf{g}_i$  is no longer equal to  $\mathbf{g}_i'$  and the corresponding  $L_i$  may not pass through the origin. As a result, there is a perpendicular distance  $d_i$  between  $L_i$  and the origin:

$$\begin{aligned} d_i &= \frac{|\mathbf{g}_i' \times ([0 \ 0 \ 0]^T - \mathbf{h}_i)|}{|\mathbf{g}_i'|} \\ &= \frac{|\mathbf{g}_i' \times (-\mathbf{h}_i)|}{|\mathbf{g}_i'|} \\ &= \sqrt{\frac{(h_z^i V_i' - h_y^i F)^2 + (h_x^i F - h_z^i U_i')^2 + (h_y^i U_i' - h_z^i V_i')^2}{U_i'^2 + V_i'^2 + F^2}} \end{aligned} \quad (3.14)$$

As there are  $N_m$  feature points, there is a set of perpendicular distance  $\mathbf{D} = \{d_1, d_2, d_3, \dots, d_{N_m}\}$  as shown in Fig. 3. 3.

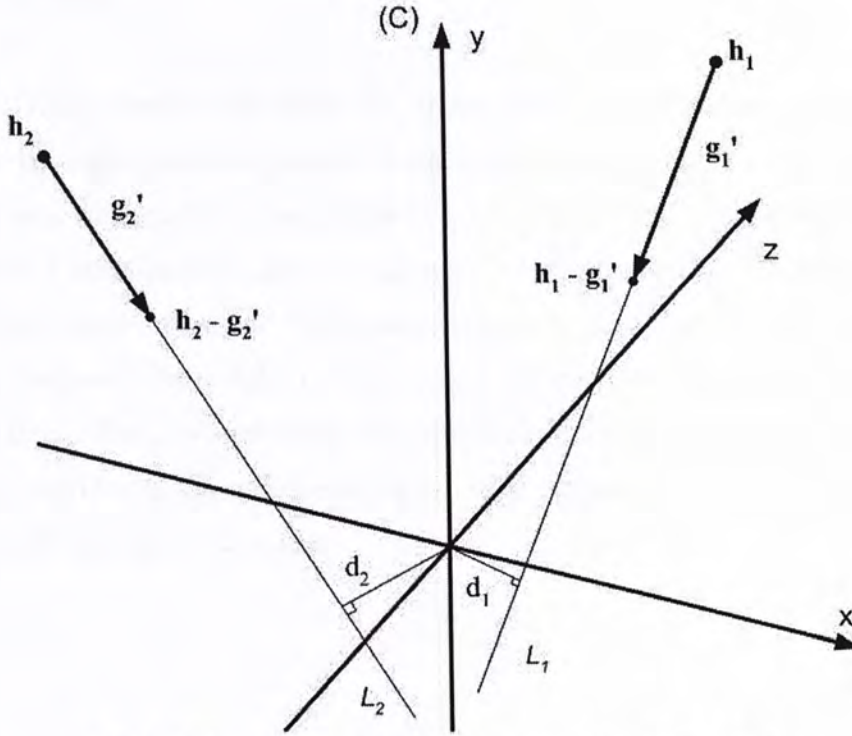


Fig. 3. 3 Visualization of the goodness of a camera state ( $N_m = 2$ ).

Hence, the accuracy of a possible camera state  $\mathbf{r}$  can be represented as the sum of  $\{d_i\}$ . The smaller the sum of  $\{d_i\}$ , the more accurate the camera state. Thus, a vision-based localization problem can be modeled as an optimization problem which minimizes the perpendicular distances between the origin and camera position lines with respect to the camera state, i.e.

$$\text{robot state} = \arg \min_{\mathbf{r}} \sum_{i=1}^{N_m} d_i(\mathbf{r}) \quad (3. 15)$$

Thus, the goodness of a possible camera state can be measured by the following function:

---

$$f_{goodness}(\mathbf{r}) = \sum_{i=1}^{N_m} d_i(\mathbf{r}) \quad (3.16)$$

### 3.3 Summary

The vision-based localization for a high DOFs moving robot is defined as: given the landmark positions and the feature coordinates from an image captured at a certain time step, the 6D camera state (i.e. 3D position and 3D orientation) related to the world coordinate system at that time step is estimated. We formulate the localization problem as an optimization which minimizes the perpendicular distances between the origin of the camera coordinate system and the camera position lines. Thus, by optimizing the objective function, the camera state can be obtained. In addition, the transformation between the estimated camera state and the estimated robot state is presented.

---

## ***Chapter 4 – Existing Search Algorithms***

In chapter 3, we showed that the vision-based localization problem can be formulated as an optimization in which the camera state at current time step can be obtained by minimizing the objective function  $f_{goodness}(\mathbf{r})$  (Eq. (3. 16)). There are numerous search methods proposed, in which different approach advantages in finite classes of functions. In the rest of this chapter, a brief discussion of the common search methods is given. Afterward, the choice of search algorithm for the proposed objective function is analyzed.

### **4.1 Overview of the Existing Search Algorithms**

In numerous search algorithms, there are three major approaches: calculus-based, enumerative and stochastic. The calculus-based approach can be further divided into two groups: direct and indirect methods. The direct search method searches for the local optimum by constantly adjusting the solution in the search space according to the gradient information. A well-known method in this domain is the steepest gradient descent. For the indirect method, the local optimum is found by solving a set of equations obtained by setting the gradient of the objective function in each direction to zero. The calculus-based search method is efficient as it is specialized for different objective functions. However, as it seeks for optimum based on the neighborhood information, the search result may probably be trapped in the local optimum for multimodal functions. Moreover, the arbitrarily chosen step size of the algorithm may result in divergence. In addition, the existence of derivatives of the objective function is necessary. Therefore, the calculus-based search method is suitable only for unimodal and differentiable objective functions.

For the enumerative method, its principle is simple, which searches for the global optimum by evaluating every point in the search space, one at a time. Though this approach ensures that the global optimum can be found, it is inefficient and is only applicable on small search space and discounted processing time.

For the stochastic search algorithm, it is not just randomly searching in a search space. Instead, it uses randomized techniques to guide the searching process. It is not strange to apply random in the searching process as many processes in our nature involve random. A famous stochastic search algorithm is called the genetic algorithm (GA), which employs the mechanism of natural selection in its search strategy. Stochastic algorithm is not as efficient as the calculus-based approach. However, since no gradient information of an objective function is involved in GA, it can be applied to both discontinuous and non-differentiable functions. Moreover, as the GA search is guided by a random operator instead of the gradients of function, it is able to jump out the local optimum lobe.

The three search approaches: calculus-based, enumerative and stochastic, benefit in different classes of functions and with different levels of efficiency. The calculus-based method is the most efficient solution among the rest but its application is limited to differentiable functions. Besides, its performance is not stable for multimodal functions as it is probably trapped in local optimum. Though the enumerative method is inefficient, it ensures that the global optimum can be found and is applicable to both non-differentiable and multimodal functions. Moreover, its parameter-less property is the advantage over the other two approaches. Though the stochastic search approach is not as efficient as the calculus-based approach in differentiable and unimodal functions, it is more robust in the sense that its performance is stable for wider classes of functions including multimodal, non-differentiable and discontinuous functions. The stochastic search approach is not doing better than the enumerative approach when unlimited time is given. However, it is a more efficient and is a fairly good solution when processing time is critical.

## 4.2 Search Algorithm for the Proposed Objective Function

In chapter 3, we pointed out that the error of a suggested robot state  $\mathbf{r}$  in vision-based localization can be expressed as  $f_{goodness}(\mathbf{r})$ , i.e.  $\mathbf{r}_1$  is more accurate than  $\mathbf{r}_2$  if  $f_{goodness}(\mathbf{r}_1) < f_{goodness}(\mathbf{r}_2)$ . Thus, the camera state can be obtained by minimizing the objective function. In calculus-based approach, we should set partial gradient of



$f_{goodness}(\mathbf{r})$  to zero. On the other hand,  $f_{goodness}(\mathbf{r})$  is a sum of the perpendicular distances  $d_i$  between  $L_i$  and origin at the camera coordinate system  $C$ . As  $d_i$  in Eq. (3.14) consists of trigonometry functions (sine and cosine), these non-linear terms are not eliminated after the partial differentiation in the calculus-based approach. Instead, more non-linear terms are introduced that the resultant differential equations are much higher non-linear. Due to the non-linearity, the matrix inverse method is not applicable. Meanwhile, this non-linearity of  $f_{goodness}(\mathbf{r})$  implies that  $f_{goodness}(\mathbf{r})$  is multimodal.

Fig. 4. 1 shows the simplified output landscape of  $f_{goodness}(\mathbf{r})$  when number of feature points  $N_m = 5$  and the feature points are viewed at the position = [-1.77, 29.22, 78.00] and the orientation = [-0.07°, 0.04°, -0.03°] represented by the Euler angles. In this figure,  $f_{goodness}(\mathbf{r})$  is plotted against  $p_x$  and  $\theta_y$  (indicated as “angle y” in the figure) while  $p_y$ ,  $p_z$ ,  $\theta_x$  and  $\theta_z$  are fixed at the desired values (i.e.  $p_y = 29.22$ ,  $p_z = 78.00$ ,  $\theta_x = -0.07^\circ$  and  $\theta_z = -0.03^\circ$ ).

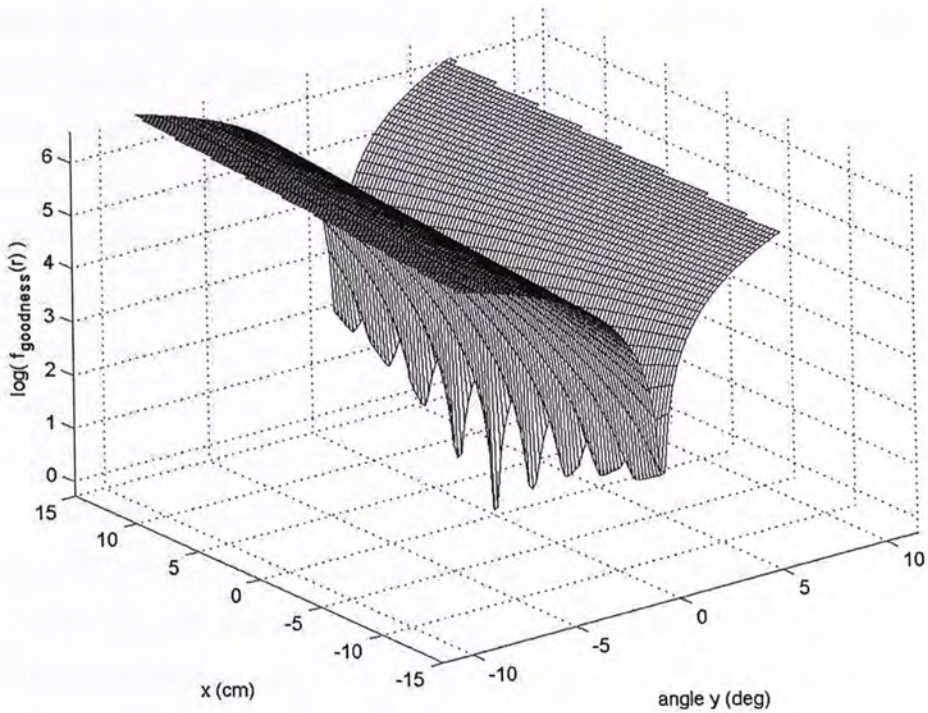


Fig. 4. 1 The objective function  $f_{goodness}(\mathbf{r})$  is multimodal.

Seen from the figure, the landscape contains multiple local optimums and a single global optimum lobe (the one we are interested in). It is expected that there will be even more local optimums when all the elements of  $\mathbf{r}$  are variable. Therefore, the steepest gradient descent method and its variants are easily trapped in the local optimum. For the enumerative approach, it is inefficient to optimize  $f_{goodness}(\mathbf{r})$  in the following aspects: 1) fast algorithm is desired for robot localization, otherwise the system will lose track of the robot and 2) the search space is large, due to the high-dimensional camera state. Lastly, the stochastic search approach is considerable in optimizing  $f_{goodness}(\mathbf{r})$  as it is relatively efficient and performs well on multimodal functions, and GA is employed in the proposed algorithm.

### 4.3 Summary

The common search algorithms can be classified into three types: calculus-based, enumerative and stochastic. In the view of efficiency, the indirect method in the calculus-based approach (i.e. matrix inverse method) is the most efficient approach because it is specialized for the objective function. The optimum is obtained by setting the gradients of the function to zero. The enumerative method is the most insufficient approach as it evaluates every point in the search space. Comparing the limitations of these algorithms, the calculus-based method is practical to differentiable functions only. Moreover, since the calculus-based method is guided by the gradients information, it is easily trapped in the local optimum for multimodal functions. The enumerative method is limited to small search space. Moreover, it provides a discrete solution with the resolution depends on the sampling rate. The efficiency of the enumerative method is inversely proportional to the resolution in the order of function dimension. The stochastic method is applicable to all classes of functions including: discontinuous, non-differentiable, multimodal and even black-box functions. It is not limited by the size of the search space. The limitations and efficiency of these search algorithms are summarized in Table 4.1 and Table 4.2 respectively.

	<i>Calculus-based</i>	<i>Enumerative</i>	<i>Stochastic</i>
<i>Continuous functions</i>	yes	no	no
<i>Unimodal functions</i>	yes	no	no
<i>Differentiable functions</i>	yes	no	no
<i>Small search space</i>	no	yes	no

Table 4.1 Limitations of the calculus-based, enumerative and stochastic approaches.

	<i>Calculus-based</i>	<i>Enumerative</i>	<i>Stochastic</i>
<i>Differentiable and unimodal function</i>	high	low	moderate
<i>Differentiable and multimodal function</i>	--	low	moderate
<i>Non-differentiable and unimodal function</i>	--	low	moderate
<i>Non-differentiable and multimodal function</i>	--	low	moderate

Table 4.2 Efficiency of the calculus-based, enumerative and stochastic approaches.

## **Chapter 5 – Proposed Vision-based Localization using Genetic algorithm**

The formulation of the vision-based localization for high degree of freedom (DOFs) robot as an optimization problem is discussed in chapter 3. In this chapter, the details of optimizing the objective function by the genetic algorithm (GA) are discussed. In the rest of this chapter, we first present the mechanism of the genetic algorithm. Afterward, we discuss the details of the employment of the genetic algorithm including chromosome formation, fitness function, genetic operators, selection scheme and search space.

### **5.1 Mechanism of Genetic Algorithm**

Genetic algorithms (GAs) are search algorithms model the mechanism of natural selection. A candidate solution of an optimization problem is represented as a string of elements called chromosome, where each element (gene) is one of the parameters in the solution. The length of a chromosome equals to the function dimension. For example, in the optimization of a function  $f(x_1, x_2, x_3, \dots, x_n)$ , the chromosome should be in the form as shown in Fig. 5. 1.

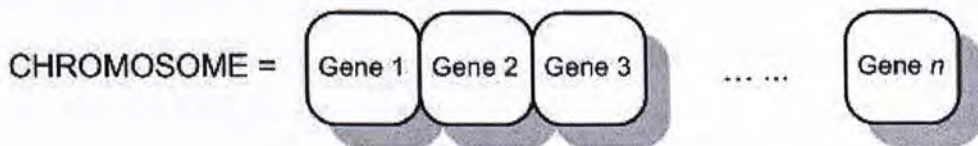


Fig. 5. 1 Chromosome and genes.

In the initialization of GA, a set of chromosomes (population pool) are generated randomly within the search space, where the number of chromosomes in a pool is known as population size. After the initialization, GA performs the search by means of reproducing new chromosomes (offspring/ children) from the initial population (parents) through the genetic operators. Two common genetic operators are mutation and crossover. Mutation reproduces a child by varying the values of

some genes in a selected parent. Fig. 5. 2 shows an example of mutation. The selection of genes to mutate and the amount to mutate are governed by a random process. As the mutation generated offspring is slightly different from its parent, they correspond to the local search in the searching process.

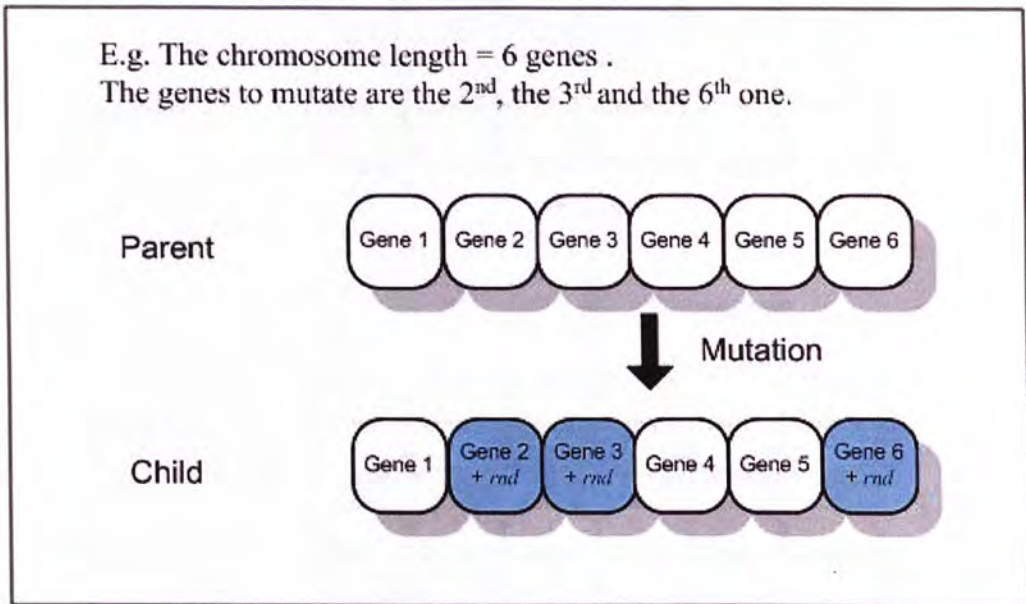


Fig. 5. 2 The mutation process.

Given two parents, crossover reproduces two children by cutting the parents into segments and swapping some of them between the two parents. Fig. 5. 3 shows an example of crossover. The decisions of cutting position and the segments to be swapped are governed by a random process. The crossover generated offspring are largely different from their parents. Hence, they correspond to the far search in the searching process.

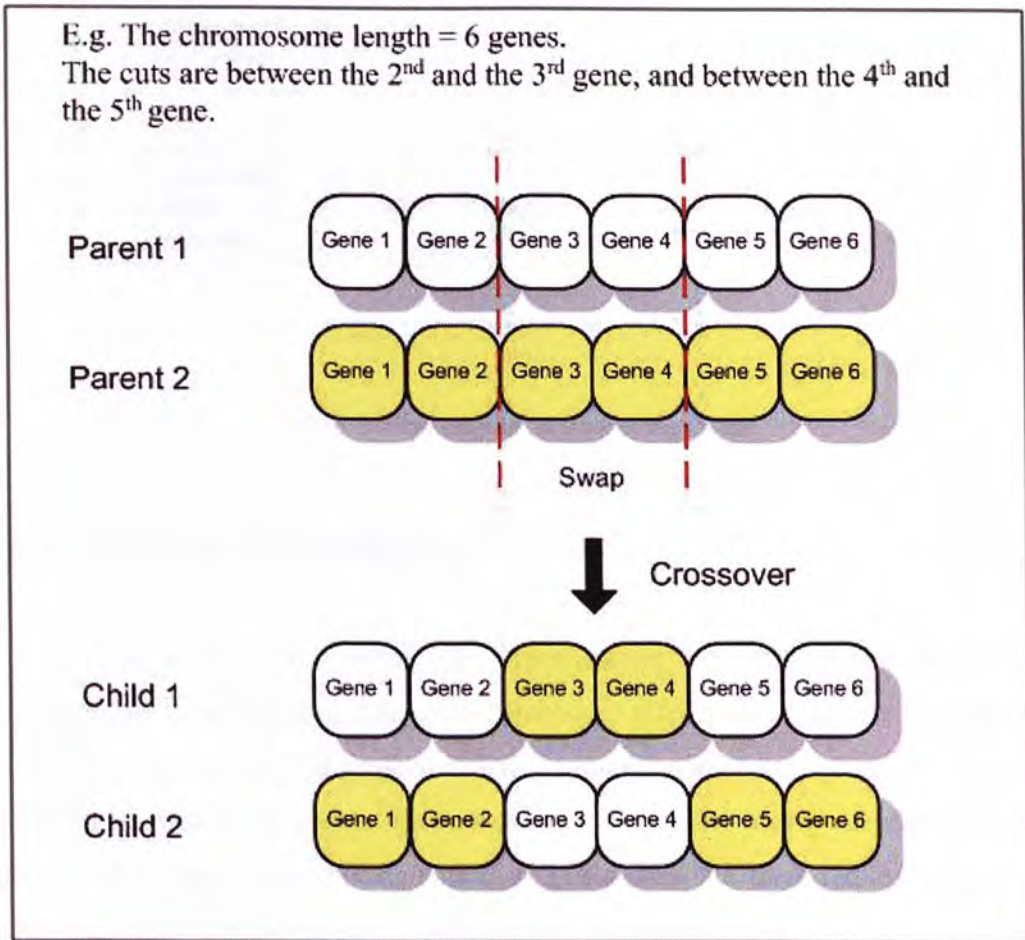


Fig. 5. 3 The crossover process.

The goodness (fitness) of the parents and offspring are measured by a fitness function which is the function to be optimized. A set of chromosomes with size equal to the population size is selected from the mixture of parents and offspring by means of “survival of the fittest”. The selected chromosomes become the parents in the next generation. This is known as the selection process. The reproduction and selection processes are repeated until the fitness of the chromosomes are converged or the generation number exceeds a certain threshold. The fitness of the chromosomes is expected to be improved along the generations. The most optimal chromosome found in the evolution is regarded as the solution of the optimization. The flow diagram of general GA is shown in Fig. 5. 4.

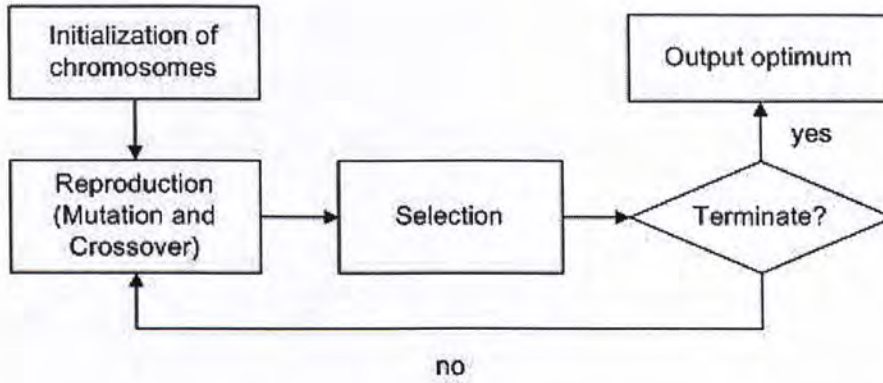


Fig. 5. 4 Flow diagram of genetic algorithm.

## 5.2 Formation of Chromosome

In this section, we construct the chromosome with respect to our proposed objective function. The formation of chromosome affects the searching efficiency of a GA. The chromosome is preferred that: 1) the corresponding genes are mutually independent, which prevents the condition checks for every newly generated chromosomes and 2) the representation of genes does not have redundancy, which reduces as many optimums in the search space as possible.

The camera state: the target solution of the localization problem is modeled to be chromosome. The camera position vector  $\mathbf{p}$  is composed of three parameters:  $p_x$ ,  $p_y$  and  $p_z$  which are coordinates along the  $x$ -,  $y$ - and  $z$ - axis of the world coordinate system  $W$ . These parameters are mutually independent and the representation does not have redundancy. Therefore, they are directly defined as position genes in the chromosome, i.e.  $(P_x P_y P_z)$ .

In general, three parameters are sufficient to represent an orientation in a 3D space. As discussed in section 3.2, the camera orientation is represented by Euler angles:  $\mathbf{o} = [\theta_x, \theta_y, \theta_z]$  which describe the rotations about the  $x$ -,  $y$ - and  $z$ - axis of  $W$ . Though the camera orientation can be represented by three parameters, we model the orientation genes based on the quaternion representation which involves four parameters. The reason is that for a certain orientation, it can be expressed by more than one sets of Euler angles. Consider the following example (Fig. 5. 5(a)): a unit vector  $[0, 0, 1]^T$  is rotated according to a set of angles:  $[45^\circ, 90^\circ, -45^\circ]$  based on the right hand rule and the resultant vector is  $[-1, 0, 0]^T$ . On the other hand, this rotation

can be represented by another two sets of angles:  $[0^\circ, 90^\circ, 0^\circ]$  and  $[90^\circ, 0^\circ, -90^\circ]$  as shown in Fig. 5. 5(b) and Fig. 5. 5(c). Therefore, the rotation represented in  $[45^\circ, 90^\circ, -45^\circ]$ ,  $[0^\circ, 90^\circ, 0^\circ]$  and  $[90^\circ, 0^\circ, -90^\circ]$  are equivalent.

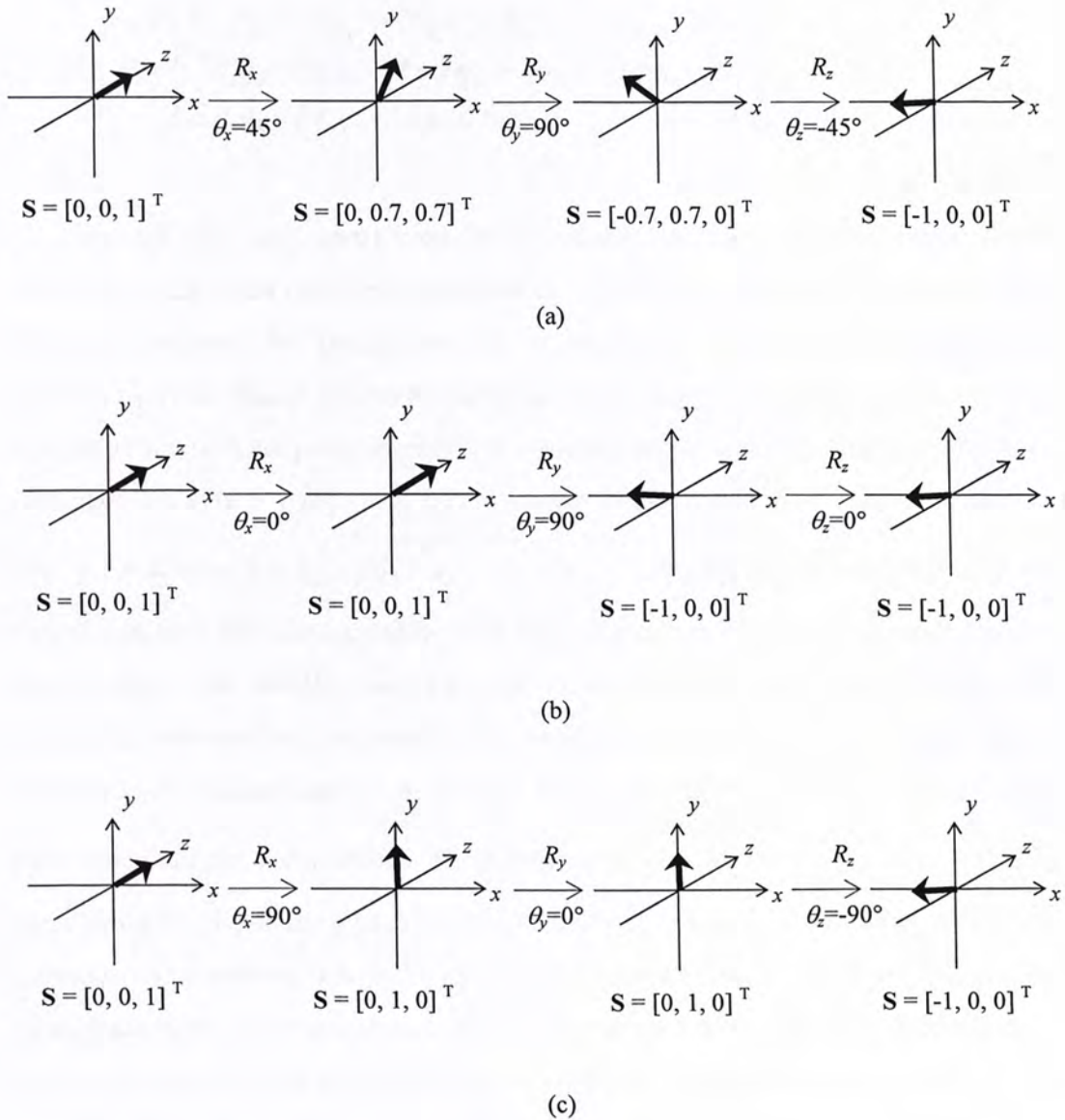


Fig. 5. 5 Example of multiple expressions of a rotation in Euler angles.

The multiple expressions in Euler angles lead to multiple optimums in the search space that waste the searching power. Consider the previous example again, if  $\theta_x$ ,  $\theta_y$  and  $\theta_z$  are defined as three orientation genes and  $[45^\circ, 90^\circ, -45^\circ]$  is an optimal orientation (i.e. either global or local),  $[0^\circ, 90^\circ, 0^\circ]$  and  $[90^\circ, 0^\circ, -90^\circ]$  are also the optimal orientations. Hence, instead of Euler angles, the orientation genes



are modeled based on the unit quaternion:  $\mathbf{q} = [q_1, q_2, q_3, q_4]^T$  where  $q_1^2 + q_2^2 + q_3^2 + q_4^2 = 1$ , which gives unique rotation representation. The rotation matrix  $\mathbf{R}$  that relates  $W$  and  $C$  can be converted from  $\mathbf{q}$  by:

$$\mathbf{R} = \begin{bmatrix} 1 - 2q_3^2 - 2q_4^2 & 2q_2q_3 - 2q_1q_4 & 2q_2q_4 + 2q_1q_3 \\ 2q_2q_3 + 2q_1q_4 & 1 - 2q_2^2 - 2q_4^2 & 2q_3q_4 - 2q_1q_2 \\ 2q_2q_4 - 2q_1q_3 & 2q_3q_4 + 2q_1q_2 & 1 - 2q_2^2 - 2q_3^2 \end{bmatrix} \quad (5.1)$$

Though the unit quaternion representation of camera orientation avoids redundancy, the extra parameter increases the number of dimension of search space. We either increase the population size or number of generation to maintain the searching power. Based on the property  $|\mathbf{q}|=1$ , we suggest to reduce the orientation representation to three parameters while avoiding the redundancy. Suppose any three parameters in  $\mathbf{q}$  (e.g.  $q_2, q_3$  and  $q_4$ ) are known, the magnitude of the remaining one (e.g.  $q_1$ ) is known, i.e.  $|q_1| = \sqrt{1 - q_2^2 - q_3^2 - q_4^2}$ . The sign of  $q_1$  is considered in the fitness function by substituting  $q_1$  with both signs, one at a time. Among the two fitness values, the smaller one is chosen as the resultant fitness (see section 5.3). Hence, the orientation representation is simplified to  $\mathbf{q}_s = [q_2, q_3, q_4]^T$  where  $|\mathbf{q}_s| \leq 1$ . However, the parameters in  $\mathbf{q}_s$  cannot be converted to genes directly as these parameters are not independent. They are related by  $\sqrt{q_2^2 + q_3^2 + q_4^2} \leq 1$ . It is time consuming to check the genes of every newly generated chromosome and to regenerate chromosomes when the condition is not satisfied. Therefore, using these three parameters as genes is also not a suitable choice. As the magnitude of  $\mathbf{q}_s$  is smaller than one, it can be regarded as a point within the unit sphere centered at the origin with radius equal to one. To define a point within a unit sphere, three independent parameters:  $\alpha, \beta$  and  $l$  can be used as shown in Fig. 5. 6, where  $\alpha$  and  $\beta$  are the pan and tilt angle respectively and  $l$  is distance from the origin ( $l \leq 1$ ).

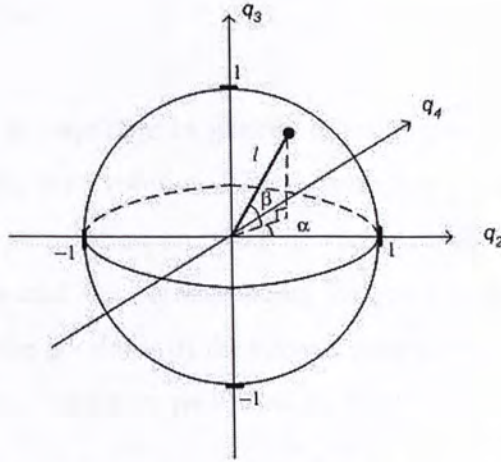


Fig. 5. 6 Representation of a point in a unit sphere using  $\alpha$ ,  $\beta$  and  $l$ .

$\mathbf{q}_s$  is related to  $[\alpha, \beta, l]$  by the following equations:

$$\begin{aligned} q_2 &= l \cdot \cos(\alpha) \cdot \cos(\beta) \\ q_3 &= l \cdot \sin(\beta) \\ q_4 &= l \cdot \sin(\alpha) \cdot \cos(\beta) \end{aligned} \quad (5.2)$$

Therefore,  $\alpha$ ,  $\beta$  and  $l$  are defined as the orientation genes, i.e.  $(Q_\alpha Q_\beta Q_l)$ . The chromosome is composed of the position and orientation genes in the form of:  $[P_x, P_y, P_z, Q_\alpha, Q_\beta, Q_l]$ .

The robot state vector  $\mathbf{r}$  is defined in Eq. (3. 9) as  $\mathbf{r} = [\mathbf{p}^T, \mathbf{o}^T]^T$ , where  $\mathbf{o}$  is the camera orientation vector represented by Euler angles. Due to the redundancy of Euler angles representation, we represent the camera orientation by unit quaternion and redefine the camera orientation vector as  $\mathbf{q} = [q_1, q_2, q_3, q_4]^T$ . Thus, the robot state vector is also redefined as:

$$\mathbf{r} = \begin{bmatrix} \mathbf{p} \\ \mathbf{q} \end{bmatrix} = [p_x \ p_y \ p_z \ q_1 \ q_2 \ q_3 \ q_4]^T \quad (5.3)$$

### 5.3 Fitness Function

Fitness function is important in genetic algorithm as it measures the qualities of chromosomes during the evolution. The formulation of fitness function depends on the nature of the optimization problem. In vision-based localization, given a set of landmark positions and the corresponding feature coordinates, the fitness of a chromosome denotes the goodness of the camera state represented by its genes.

As the robot state vector is redefined to be  $\mathbf{r} = [\mathbf{p}^T, \mathbf{q}^T]^T$  in Eq. (5. 3), we introduce a new objective function  $f_{goodness2} \left( \begin{bmatrix} \mathbf{p} \\ \mathbf{q} \end{bmatrix} \right)$  by replacing the orientation vector  $\mathbf{o}$  in  $f_{goodness} \left( \begin{bmatrix} \mathbf{p} \\ \mathbf{o} \end{bmatrix} \right)$  by  $\mathbf{q}$ . Both of  $f_{goodness2} \left( \begin{bmatrix} \mathbf{p} \\ \mathbf{q} \end{bmatrix} \right)$  and  $f_{goodness} \left( \begin{bmatrix} \mathbf{p} \\ \mathbf{o} \end{bmatrix} \right)$  are aim at measuring the sum of perpendicular distances between the origin of the camera coordinate system and the camera position lines. The only difference between these functions is the orientation representation. With the camera orientation represented by  $\mathbf{o}$ , the rotation matrix  $\mathbf{R}$  is converted according to Eq. (3. 10). On the other hand, with the camera orientation represented by  $\mathbf{q}$ ,  $\mathbf{R}$  is converted according to Eq. (5. 1).

Hence, the fitness function is formulated based on the objective function  $f_{goodness2} \left( \begin{bmatrix} \mathbf{p} \\ \mathbf{q} \end{bmatrix} \right)$ . The parameters in vector  $\begin{bmatrix} \mathbf{p} \\ \mathbf{q} \end{bmatrix}$  are mapped to the genes ( $P_x P_y P_z, Q_\alpha Q_\beta Q_l$ ) as follows:

$$\begin{aligned}
 p_x &= P_x \\
 p_y &= P_y \\
 p_z &= P_z \\
 q_2 &= Q_l \cos(Q_\alpha) \cos(Q_\beta) \\
 q_3 &= Q_l \sin(Q_\beta) \\
 q_4 &= Q_l \sin(Q_\alpha) \cos(Q_\beta) \\
 q_1 &= \pm \sqrt{1 - q_2^2 - q_3^2 - q_4^2} = \pm \sqrt{1 - Q_l^2}
 \end{aligned} \tag{5. 4}$$

There are two possible values of  $q_l$  that are opposite signs. They are substituted to the objective function one at a time. The minimum among the two resultant function values is regarded as the fitness of the chromosome. Hence, the fitness function is formulated as the following:

$$f_{fitness}([P_x, P_y, P_z, Q_\alpha, Q_\beta, Q_l])$$

$$= \min(f_{goodness2} \left( \begin{bmatrix} P_x \\ P_y \\ P_z \\ +\sqrt{1-Q_l^2} \\ Q_l \cos(Q_\alpha) \cos(Q_\beta) \\ Q_l \sin(Q_\beta) \\ Q_l \sin(Q_\alpha) \cos(Q_\beta) \end{bmatrix} \right), f_{goodness2} \left( \begin{bmatrix} P_x \\ P_y \\ P_z \\ -\sqrt{1-Q_l^2} \\ Q_l \cos(Q_\alpha) \cos(Q_\beta) \\ Q_l \sin(Q_\beta) \\ Q_l \sin(Q_\alpha) \cos(Q_\beta) \end{bmatrix} \right)) \quad (5.5)$$

#### 5.4 Mutation and Crossover

The GA performs the optimum search by means of reproduction, in which new chromosomes are generated using genetic operators. Two standard genetic operators: mutation and crossover are used in the proposed algorithm.

In the crossover operation, two chromosomes (parents) are randomly selected from the current population. By cutting the chromosomes into segments and swapping them between the two parents, two children are reproduced. In the proposed algorithm, multi-point crossover is used where each segment is one gene long. For each pair of selected parents, a swapping vector  $[s_1, s_2, s_3, s_4, s_5, s_6]$  is generated, in which  $s_i, i \in [1, 6]$  is a Boolean random variable.  $s_i = 1$  indicates that the  $i^{th}$  genes in the parents are swapping, while  $s_i = 0$  indicates that no swap is performed in the  $i^{th}$  gene. For example, if the swapping vector for a pair of parents is  $[1, 0, 0, 1, 1, 0]$ , two individuals are generated as shown in Fig. 5. 7.

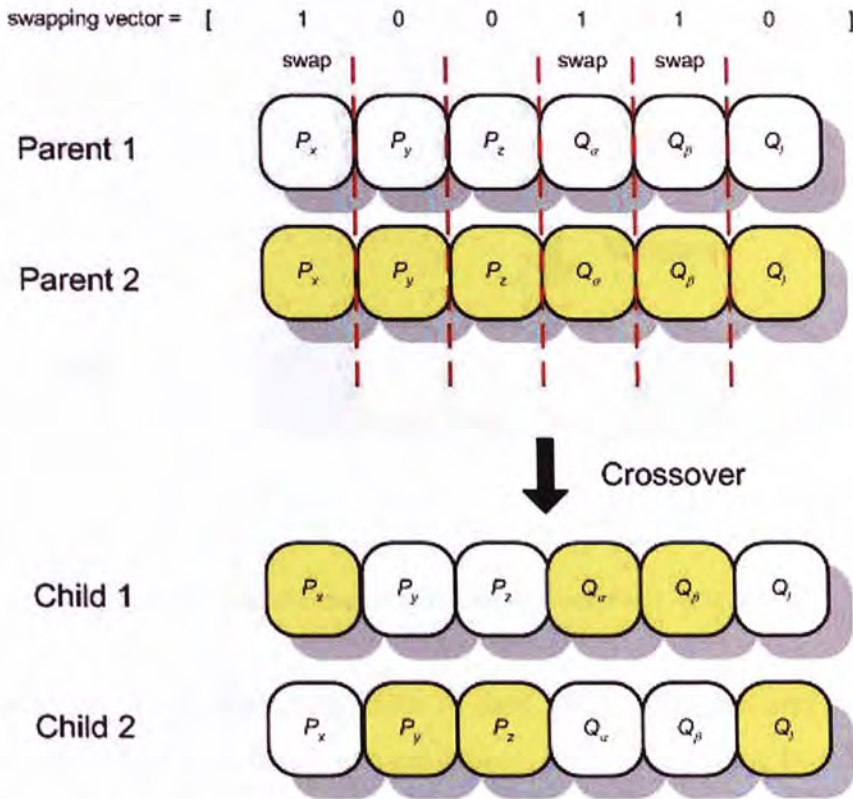


Fig. 5. 7 Example of crossover with the swapping vector = [1, 0, 0, 1, 1, 0].

Hence, the effective crossover rate of each gene is 0.5. In a generation,  $N_p$  individuals are generated by crossover, where  $N_p$  is the population size. Since the offspring generated from crossover are significantly different from its parents, it facilitates the far search in the searching process.

Mutation is another genetic operator. Given a chromosome (parent) from the current population, a new individual is reproduced by mutating some of the genes in the parent. Similar to crossover process, a mutating vector  $[m_1, m_2, m_3, m_4, m_5, m_6]$  is generated for each parent, in which  $m_i, i \in [1, 6]$  is a Boolean random variable.  $m_i = 1$  represents that the  $i^{th}$  gene of the parent is mutated, while  $m_i = 0$  represents that the value of the  $i^{th}$  gene remains unchanged. The mutating magnitude of the  $i^{th}$  gene is a random value within the range  $[\Delta g_i, -\Delta g_i]$  under uniform distribution. We define  $\Delta g_i$  as 1/50 of the search range of the  $i^{th}$  gene. For example, if the mutating vector for a parent is [1, 0, 0, 0, 1, 0], the individual generated is in the form as shown in Fig. 5. 8.

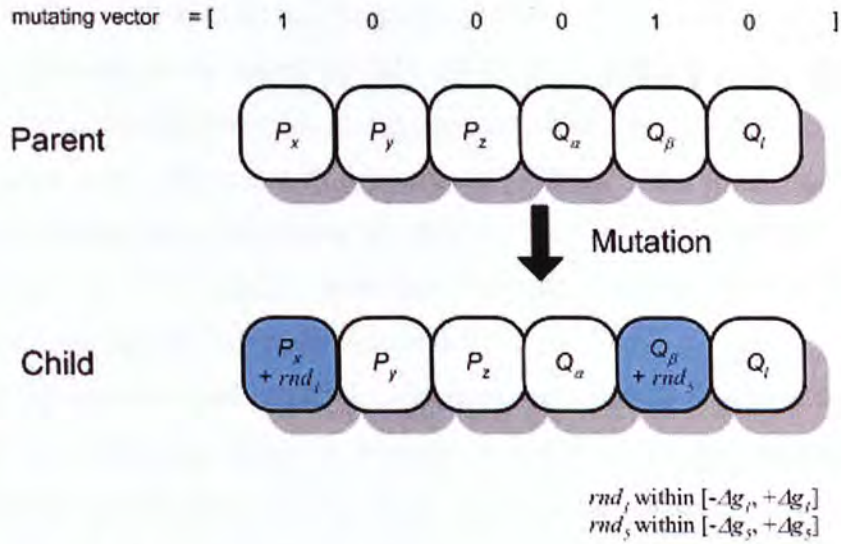


Fig. 5. 8 Example of mutation with mutating vector = [1, 0, 0, 0, 1, 0].

Therefore, the mutating magnitude of each gene is different and the effective mutation rate of each gene is 0.5. In a generation, each chromosome from the current population reproduces one individual by mutation from which  $N_p$  mutation generated individuals are formed. As  $\Delta g_i$  is small relative to the search range, the offspring generated from mutation are slightly different from its parent. Hence, it corresponds to the local search in the searching process.

## 5.5 Selection and Stopping Criteria

After the reproduction process, there are  $2N_p$  newly generated chromosomes (i.e.  $N_p$  from crossover and  $N_p$  from mutation) and  $N_p$  chromosomes from current population. To keep the number of chromosomes in the next generation be equal to the population size,  $N_p$  chromosomes are selected from them by means of “survival of the fittest”: the individuals are selected according to their fitness. In order to maintain the divergence, the current population, mutation generated individuals and crossover generated individuals, are not selected together. Instead,  $N_p/2$  chromosomes are selected from the current population and mutation generated individuals, and the other  $N_p/2$  chromosomes are selected from the crossover generation individuals. For each pair of mutation generated offspring and its corresponding parent, the one with better fitness is chosen. As a result, there are  $N_p$

chromosomes chosen from all the chromosome pairs. They are then sorted by fitness and the  $N_p/2$  fittest chromosomes are selected for the next generation (see Fig. 5. 9). This prevents both chromosomes in a parent and child pair from being selected, in order to slow down the speed of convergence. On the other hand, the crossover generated chromosomes are sorted by fitness. The  $N_p/2$  fittest chromosomes are selected (see Fig. 5. 9). Hence, there are a total of  $N_p$  chromosomes selected and they become the parents in the next generation.

The reproduction and selection process are repeated in every generation until either of the following stopping criteria is satisfied: 1) the chromosome are converged that is the fitness of the fittest chromosome remains unchanged for the number of generations that equal to the convergence threshold or 2) the number of generations exceed the maximum number of generations  $T_g$ . In the proposed algorithm, the convergence threshold and  $T_g$  are set to 30 and 100 respectively.

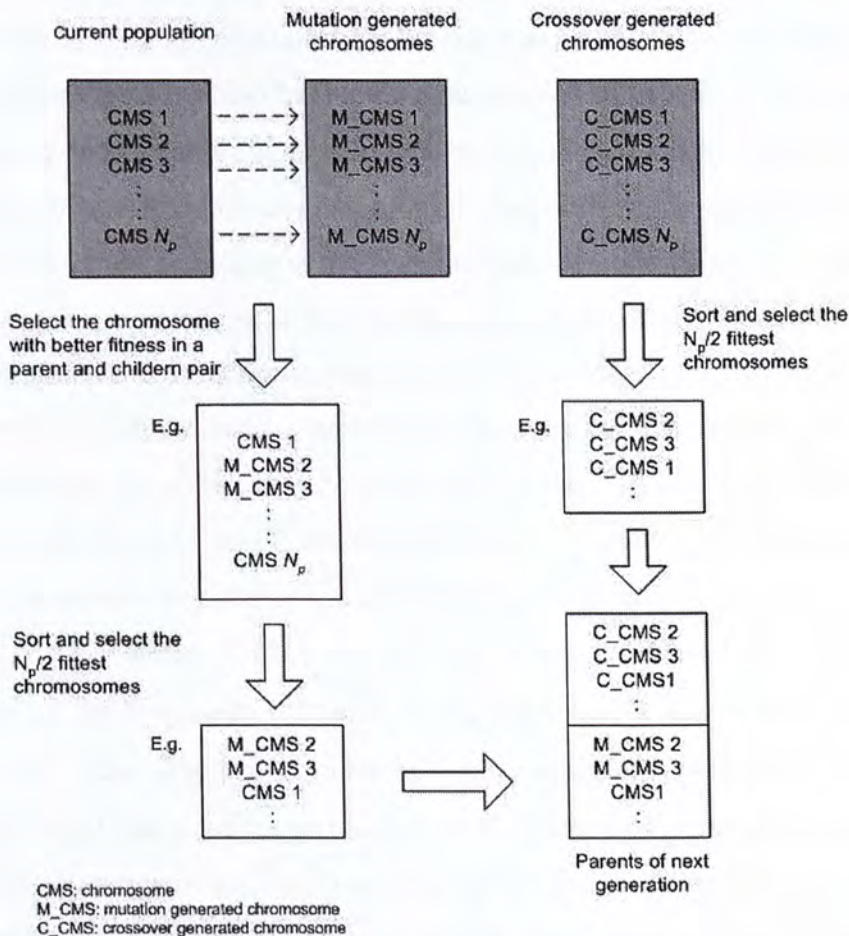


Fig. 5. 9 The selection process.

## 5.6 Adaptive Search Space

The size of search space affects the search efficiency. For the localization problem, if the search space includes all possible camera states in an environment, the search space is large that larger searching power, i.e. larger population size and/or more generations are/is needed, which increases the computational cost. However, if the search space is reduced to the camera states in part of the environment, the searching power can be saved while the actual robot state may be excluded. Therefore, we have to balance between the search space size and the probability of the actual camera state to be included.

In the proposed algorithm, we assume that the camera is allowed to rotate quickly and freely. Hence, all possible orientation states form a hyper-rectangular range  $\Omega_o = [-\pi/2, \pi/2]^2 \times [-1, 1]$ , i.e.  $[Q_\alpha, Q_\beta, Q_l] \in \Omega_o$ . Suppose the robot is walking on a floor which is parallel to the  $xz$ -plane of  $W$ , small oscillations along  $y$ -axis are occurred due to walking vibrations that the value of  $P_y$  is not always equal to  $p_{yo}$  (the camera height at steady state) but within a range, i.e  $P_y \in [p_{yl}, p_{yu}]$ , where  $p_{yl}$  and  $p_{yu}$  are the lower and upper limits of the vibrate range in  $y$ -direction. Unlike the ranges of  $[Q_\alpha, Q_\beta, Q_l]$  and  $P_y$  which are fixed for all camera state estimations, the ranges of  $P_x$  and  $P_z$  are varied according to the estimated camera states in the previous frames. This assumption is based on the fact that the current camera position should be at the neighborhood of the previous camera position unless kidnapping has happened. It is inefficient to search the camera position in the entire environment for every camera state estimation. Suppose  $t_i$  is the capture time of the  $i^{th}$  frame to process and  $\lambda_{i-1}$  is the maximum possible robot movement between  $t_i$  and  $t_{i-1}$ , the possible camera position at  $t_i$  should be within a circular range  $C_{i,1}$  centered at  $\mathbf{c}_{i-1} = [p_x^{(i-1)}, p_z^{(i-1)}]$  and radius  $\lambda_{i-1} + \Delta\lambda$  (See Fig. 5. 10) where  $[p_x^{(a)}, p_z^{(a)}]$  is the estimated  $[p_x, p_z]$  at  $t_a$  for  $a \in \mathbb{Z}^+$  and  $\Delta\lambda$  is the maximum tolerance of  $\mathbf{c}_{i-1}$ . However, if  $\mathbf{c}_{i-1}$  involves error larger than  $\Delta\lambda$ , the actual  $p_x$  and  $p_z$  at  $t_i$  may be excluded from  $C_{i,1}$ . Therefore, we propose to use the  $p_x$  and the  $p_z$  of the previous  $n > 1$  frames to define the range of  $[p_x^{(i)}, p_z^{(i)}]$ . In general, the range of  $[p_x^{(i)}, p_z^{(i)}]$  can be represented by a circular range  $C_{i,n}$  which is in terms of  $[p_x^{(i-n)}, p_z^{(i-n)}]$ , i.e.  $C_{i,n}$  centers at  $\mathbf{c}_{i-n} = [p_x^{(i-n)}, p_z^{(i-n)}]$  and radius



$\Delta\lambda + \sum_{k=1}^n \lambda_{i-k}$ . Suppose we use the estimated camera positions of the previous  $n$  frames to define the range of  $[p_x^{(i)}, p_z^{(i)}]$ , namely  $\Omega_p^{(i)}$ ,  $\Omega_p^{(i)}$  is defined as the union of  $C_{i,j}$  for  $j = 1, 2, \dots, n$ , i.e.  $\Omega_p^{(i)} = C_{i,1} \cup C_{i,2} \cup \dots \cup C_{i,n}$ . Fig. 5. 10 shows an example of  $\Omega_p^{(i)}$  for  $n = 2$ . In the proposed algorithm, we choose  $n = 3$  to define  $\Omega_p^{(i)}$  for all  $i \geq 4$ . For simplicity, the bounding box of  $\Omega_p^{(i)}$  is used as the search ranges of  $P_x$  and  $P_z$  at  $t_i$ .

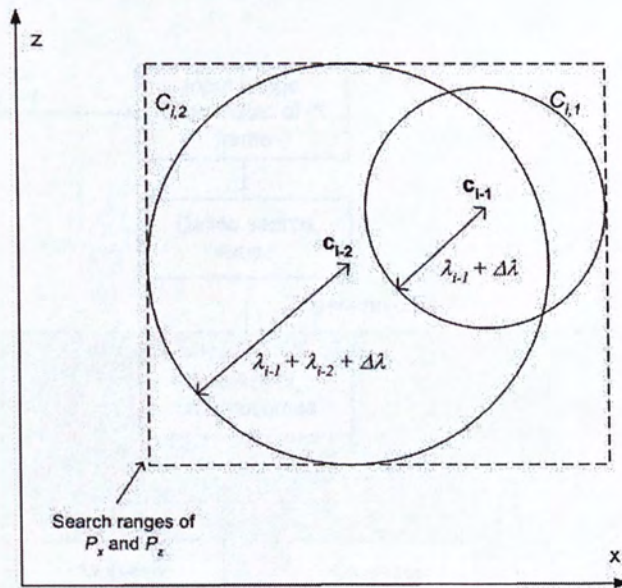


Fig. 5. 10 An example of  $\Omega_p^{(i)}$  for  $n = 2$ .

## 5.7 Overall Flow of the Proposed Algorithm

To summarize, the flow diagram of the proposed localization method using genetic algorithm is shown in Fig. 5. 11. By assuming that the chromosomes have fallen in the global optimum lobe after  $T_g/2$  generations, the efficiency of the search is increased by: 1) reducing the population size to  $N_p/2$  after the  $(T_g/2)^{\text{th}}$  generation and 2) reducing the search space to the minimum size that includes all the chromosomes at the  $(T_g/2)^{\text{th}}$  generation.

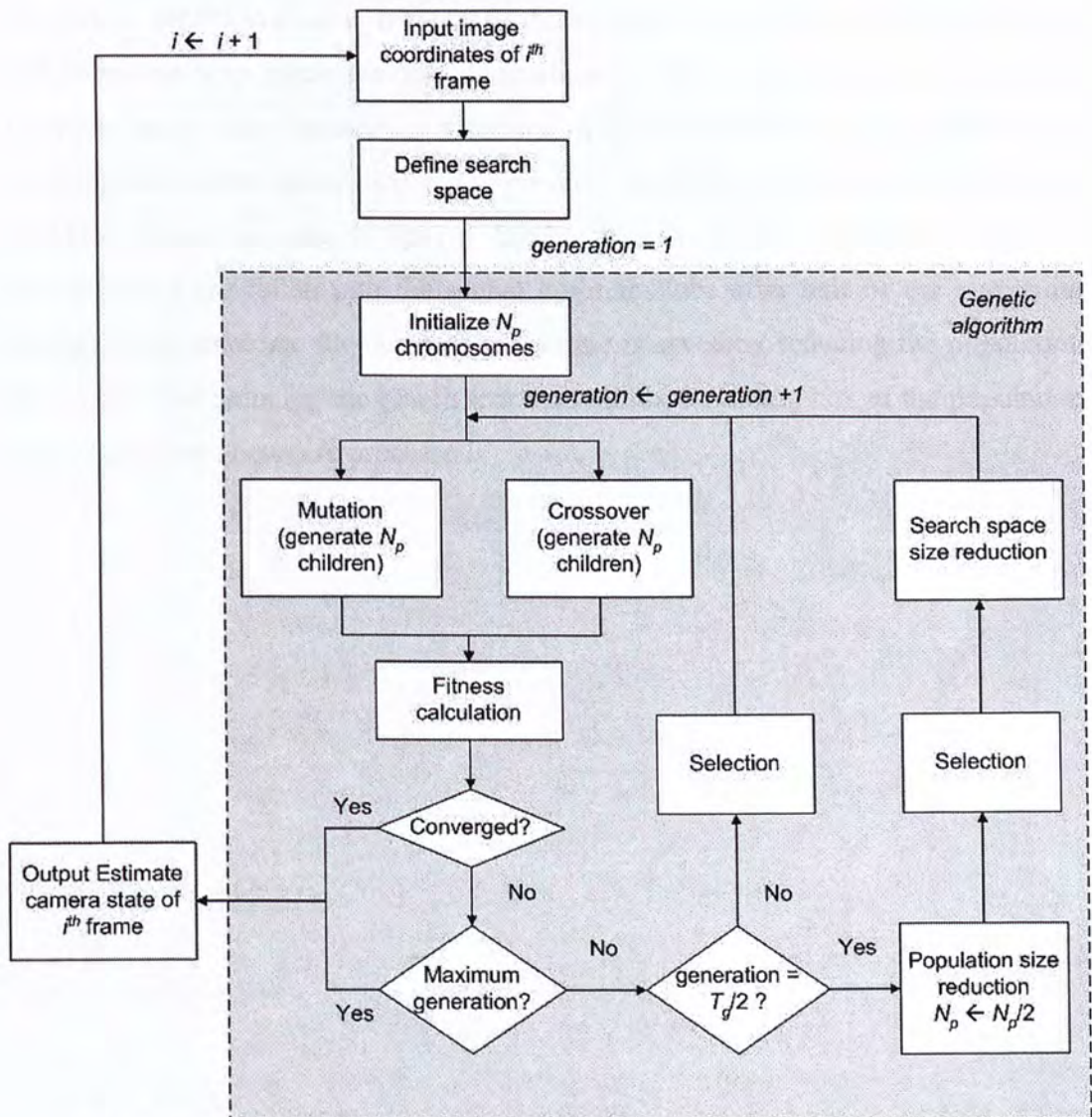


Fig. 5. 11 Flow diagram of the proposed localization algorithm using genetic algorithm.

## 5.8 Summary

In order to estimate the camera state using the landmark positions and the feature points captured from image, genetic algorithm is applied to optimize the objective function formulated in chapter 3. The details of the employment, which target the proposed objective function, are discussed. Based on the basic principle of the GA, several approaches are proposed to improve the efficiency of the localization. First, the orientation genes are constructed based on the quaternion to reduce the number of optimums. With the modification of the objective function, the orientation representation is reduced to three parameters. It increases the searching efficiency without increases the dimensions of the search space. Second, the adaptive search space strategy is proposed. It increases the searching efficiency by reducing the search space size while prevents the actual camera state from being excluded from the search space. Lastly, based on the assumption that the chromosomes are fallen into the global optimum lobe after half of the maximum number of generations, the computational time is saved by reducing the population size to half and reducing the search space size to the bounding box of the population when  $T_g/2$  generations are processed.

---

## ***Chapter 6 - Experimental Results***

In this chapter, the performance of the proposed algorithm is illustrated in terms of 1) accuracy, 2) efficiency (computational time), 3) noise sensitivity and 4) noise reduction performance by experimental results. The experiment is divided into three parts. In the first experiment, accuracy and efficiency of the proposed algorithm is examined by a simulation of continuous robot (camera) localization for 60 seconds, in which the robot is navigated in a  $10\text{m} \times 10\text{m}$  area. Afterward, the noise sensitivity of the proposed algorithm is studied in the second experiment. The setup is similar to the first experiment except that the input feature points are added with different levels of noise. In the third experiment, the noise reduction performance of the proposed adaptive search space strategy is proven by demonstrating the effect of noisy feature points on the estimated camera state when the localization is not applied with the adaptive search space strategy.

### **6.1 Test Robot**

The performance of the proposed algorithm is examined on a four-legged robot pet: Sony AIBO ERS-7 as shown in Fig. 6. 1. The robot is about 30 cm tall and 30 cm long. The robot head is equipped with a color CCD camera, whose horizontal and vertical angles of view are 56.9 and 45.2 degrees respectively. The resolution of the captured image is  $412 \times 318$  pixels and the maximum capture rate is 30 frame/sec. The images are transmitted to a networked computer via the WLAN card equipped in the robot. The frame rate of images received by the computer depends on the bandwidth of the network, which varies with time and usually cannot meet the maximum frame rate. Using the input images, the localization process is performed on the same computer. The robot movement can be remotely controlled by the control commands developed from the AIBO software development kit (SDK).



Fig. 6. 1 Test robot: Sony AIBO ERS-7.

## 6.2 Simulator

In order to illustrate the accuracy of the proposed algorithm, the estimated camera state is compared with the actual camera state. However, the three dimensional rotation of the camera is difficult to measure as the camera is equipped inside the robot head. Therefore, a simulator is developed and used in our experiments. The simulator performs three kinds of simulation: 1) camera states given the control commands, 2) vibrations caused by the oscillated walking motion of the robot and 3) input image for localization.

### 6.2.1 Camera states simulation

Given a series of control commands, the simulator simulates the walking path of the robot. Instead of providing the statuses of the controllable DOFs of the physical robot, the robot path is generated in terms of a series of discrete 6D camera states at frequency equal to the input frame rate. The simulator generated camera state  $\mathbf{r}_s$  is in the form:  $\mathbf{r}_s = [p_x, p_y, p_z, \theta_x, \theta_y, \theta_z]$ , where  $[p_x, p_y, p_z]$  is the camera position vector in Cartesian coordinates and  $[\theta_x, \theta_y, \theta_z]$  is the camera orientation vector represented by Euler angles. Both of the camera position and orientation vector are related to the world coordinate system  $W$ .

To simplify the experiment without loss of generality, we consider only the motion made by leg movement though the robot has three DOFs in the head. The commands processed by the simulator are limited to two types: 1) forward and

backward movement, e.g. “walking forward 30 cm” and 2) left and right rotation, e.g. “rotate right 30 degrees”. In this stage,  $p_x, p_z$  and  $\theta_y$  are generated with respect to the control commands and  $p_y, \theta_x$  and  $\theta_z$  are set to constant by assuming that there is no vibration caused by the walking motion, i.e.  $p_y = p_{y0}, \theta_x = 0^\circ$  and  $\theta_z = 0^\circ$ , where  $p_{y0}$  is the camera height when the robot is steady. The walking vibration is reflected by simulation described in section 6.2.2.

Compare with the movements of wheeled robot, the legged robot movements corresponds to the control commands are not precise. Moreover, leg slippages are usually happened in legged robot. Hence, in order to reflect the unmodeled movement and rotation errors, random noises are added to  $p_x, p_z$  and  $\theta_y$ . Movement/rotation error is regarded as the difference between the actual movement/rotation and the expected movement/rotation with respect to the control commands. The rotation error is simulated by adding a zero-mean Gaussian noise with standard deviation  $\sigma_r$  to the expected rotation, where  $\sigma_r$  is equal to  $10^\circ$ . The movement error is simulated by adding a zero-mean Gaussian noise with standard deviation  $\sigma_m$  to the expected movement, where  $\sigma_m$  is 10% of the expected movement.

### 6.2.2 Oscillated walking motion simulation

In the real situation, the values of  $p_y, \theta_x$  and  $\theta_z$  are not fixed as the camera vibrates due to the oscillated walking motion. The camera vibrations are simulated by the synthesis of two sources of noises: 1) the up and down vibration (movement along y-direction) is simulated by adding a random noise within the range [-3 cm, 3 cm] under uniform distribution to  $p_{y0}$  and 2) the camera orientation changes are simulated by adding random noise within the range  $[-10^\circ, 10^\circ]$  under uniform distribution to  $\theta_x, \theta_y$  and  $\theta_z$ .

### 6.2.3 Input images simulation

Given a camera state and the landmarks positions, the simulator simulates the image captured by the camera at that camera state. The image is generated in terms a set of feature coordinates. For each landmark that fall within the field of view of the camera at that camera state, the corresponding feature coordinates are generated. The simulated feature points are used as inputs of the localization system. In the case

that more than one landmarks are projected on the same image coordinates, the corresponding feature points are regarded to be distinct.

The camera states generated by the simulator are regarded as the actual camera states in the experiments. Hence, the actual camera states can be conveniently obtained and compared with the estimated camera states, in order to examine the performance of the proposed algorithm. In addition, the frequency of the generated camera states is equal to the input frame rate, which ensures that each input image has a corresponding actual camera state for comparing with the estimated camera state. Moreover, the performance of the proposed algorithm in different image noise levels can be easily investigated by adding noise to the feature coordinates using the simulator. Therefore, we perform the experiments in computer simulation instead of a physical robot.

### 6.3 Computer for simulations

All simulations are performed on the PC platform with 2.8GHz CPU and 512MB memory.

### 6.4 Position and orientation errors

In following experiments, the accuracy of an estimated camera state is represented by two quantities: position error  $E_p$  and orientation error  $E_o$ . The position error denotes the Euclidian distance between the actual and the estimated positions (3D) of the camera states. To compare the actual camera orientation represented by Euler angles (i.e.  $[\theta_x, \theta_y, \theta_z]$ ) with the estimated camera orientation represented by quaternion (i.e.  $[q_1, q_2, q_3, q_4]$ ), both of them are converted to rotation matrices, i.e. the actual rotation matrix  $R_a$  and the estimated rotation matrix  $R_e$ . Afterward, a unit vector along  $x$ -axis, i.e.  $[1, 0, 0]$ , is multiplied to both rotation matrices. The two rotated vectors are equal if the two rotation matrices are identical. If the two rotated vectors are not equal, there is an angle  $\theta_{ex} > 0$  between the two rotated vectors. The difference between the actual and estimated orientations can be described by  $\theta_{ex}$ . However, in order to avoid the problem of gimbal lock (i.e. rotating

a vector along the  $x$ -axis about the  $x$ -axis does not have effect), unit vectors along the  $y$ - and the  $z$ -axis, i.e.  $[0, 1, 0]$  and  $[0, 0, 1]$ , are also multiplied to both rotation matrices. Similarly, two angles between the two rotated vector pairs,  $\theta_{ey}$  and  $\theta_{ez}$ , were obtained. Suppose  $R_a$  and  $R_e$  are in the forms:

$$\mathbf{R}_a = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = [\mathbf{A}_1 \quad \mathbf{A}_2 \quad \mathbf{A}_3]$$

$$\mathbf{R}_e = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} = [\mathbf{B}_1 \quad \mathbf{B}_2 \quad \mathbf{B}_3]$$

,  $\theta_{ex}$ ,  $\theta_{ey}$ ,  $\theta_{ez}$  can be computed by:

$$\begin{aligned} \theta_{ex} &= \cos^{-1} (\mathbf{A}_1 \bullet \mathbf{B}_1) \\ \theta_{ey} &= \cos^{-1} (\mathbf{A}_2 \bullet \mathbf{B}_2) \\ \theta_{ez} &= \cos^{-1} (\mathbf{A}_3 \bullet \mathbf{B}_3) \end{aligned} \tag{6.1}$$

The error of an estimated orientation is defined as the average of these three angles. Since GA is a stochastic method, a simulation of the localization is repeated for many independent trials.  $E_p$  and  $E_o$  are the average over the number of processed frames in the  $N_t$  trials. Suppose  $E_p(i, n)$  and  $E_o(i, n)$  are the position error and the orientation error of the  $n^{\text{th}}$  processed frame in the  $i^{\text{th}}$  trial respectively where  $i \in [1, N_t]$ ,  $n \in [1, \eta_i]$  and  $\eta_i$  is the number of frame processed in the  $i^{\text{th}}$  trial, the  $E_p$  and  $E_o$  are represented by:

$$E_p = \frac{\sum_{i=1}^{N_t} \sum_{n=1}^{\eta_i} E_p(i, n)}{\sum_{i=1}^{N_t} \eta_i}, \tag{6.2}$$



$$E_o = \frac{\sum_{i=1}^{N_t} \sum_{n=1}^{\eta_i} E_o(i, n)}{\sum_{i=1}^{N_t} \eta_i}. \quad (6.3)$$

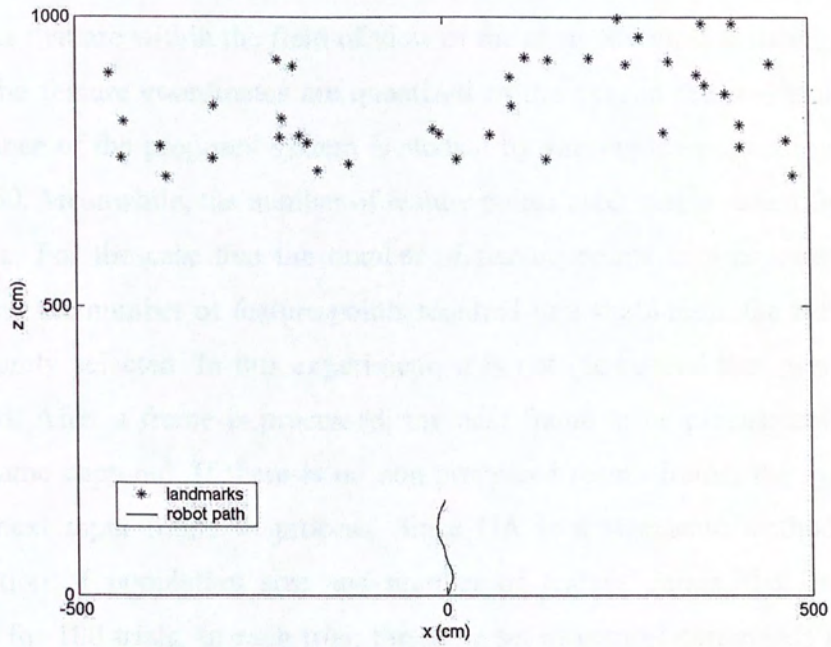
## 6.5 Experiment 1 – Feature points with quantized noise

In this experiment, the performance of the proposed algorithm is measured in terms of 1) accuracy of the estimated camera state and 2) efficiency (computational time). The feature points used in this experiment consist of quantized noise only.

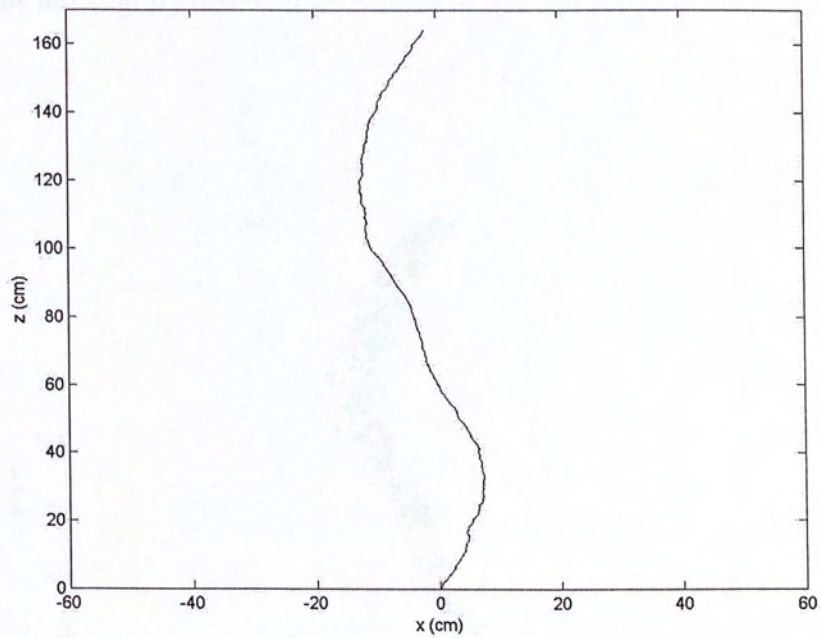
### 6.5.1 Setup

In the following simulations, the robot is navigated in an area of 1000 cm x 1000 cm consisting of 40 landmarks with known positions. The robot walks at a speed about 3 cm/sec. for 60 seconds according to a series of control commands. The magnitudes of the movement and rotation commands are within the ranges [3 cm, 6 cm] and [-10°, 10°] respectively. The initial position of the robot is assumed to be known.

By assuming that images are input to the system at a constant rate (20 frames/sec.), the simulator generates a sequence of camera states along the path at the same frequency (see section 6.2.1). Thus, 1200 camera states are generated, which are regarded as the actual camera states. Random noises are added to each camera states to synthesize the vibrations caused by the oscillated robot walking motion as described in section 6.2.2. The camera states sequence generated by the simulator forms an actual path. The landmarks distribution and an example of actual path are shown in Fig. 6. 2.



(a)



(b)

Fig. 6. 2 (a) Landmarks and actual path (Topview) and (b) magnified figure of actual path (Topview).

For each camera state, the simulator generates all the feature coordinates of landmarks that are within the field of view of the camera with that state (see section 6.2.3). The feature coordinates are quantized as the case in the real situation. The performance of the proposed system is studied by varying the population size from 100 to 300. Meanwhile, the number of feature points used is also varied from 5 to 10 per frame. For the case that the number of feature points in a captured image is larger than the number of feature points required in a simulation, the feature points are randomly selected. In this experiment, it is not guaranteed that every frame is processed. After a frame is processed, the next frame to be processed is the most recent frame captured. If there is no non-processed recent frame, the system waits for the next input frame to process. Since GA is a stochastic method, for each combination of population size and number of feature points, the simulation is repeated for 100 trials. In each trial, the same set of control commands is used and different random noises are added to the simulated camera states, in which the 100 actual paths are slightly different. A picture of the 100 actual paths is shown in Fig. 6. 3.

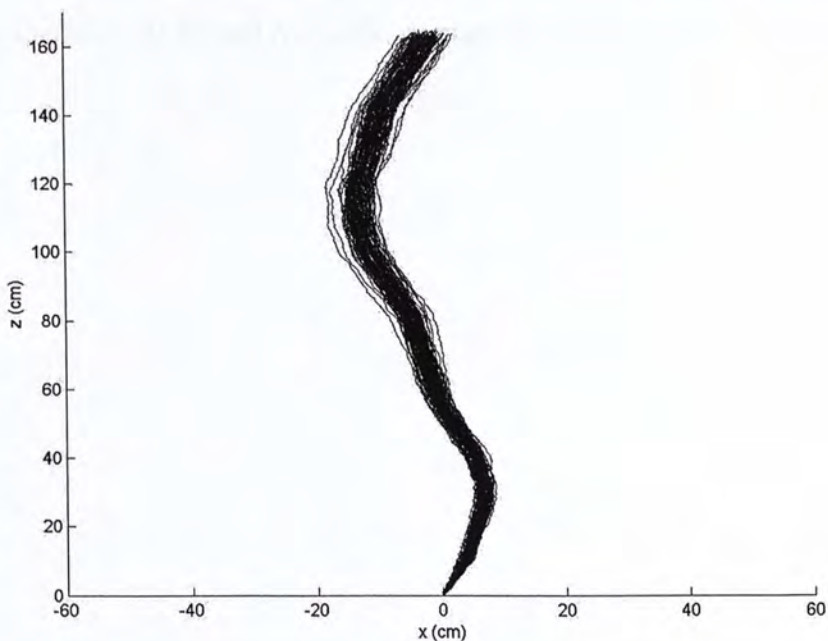
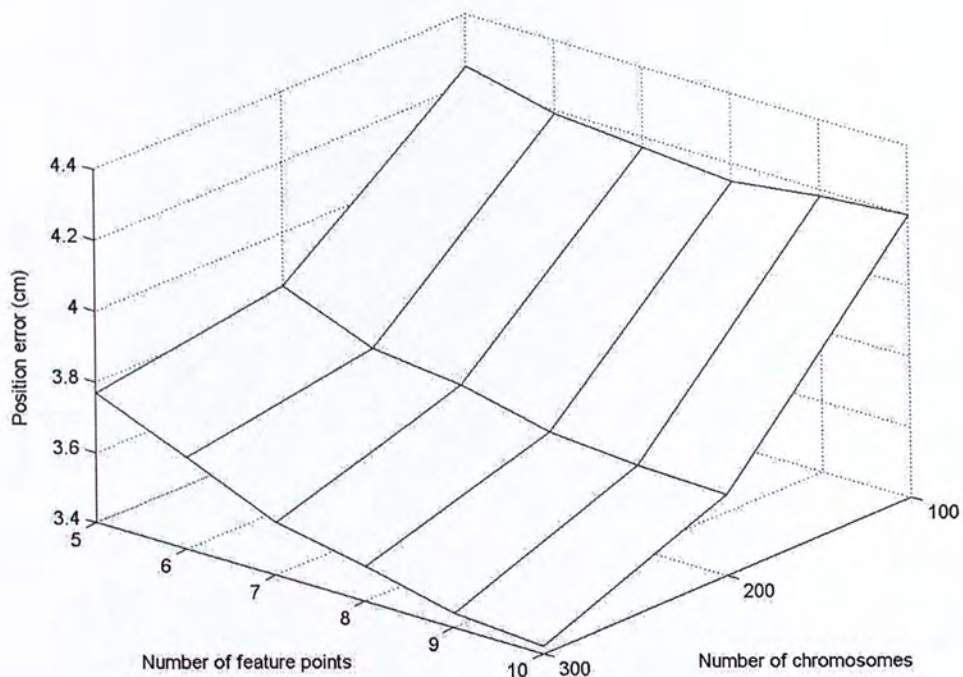


Fig. 6. 3 The actual paths of 100 trials.

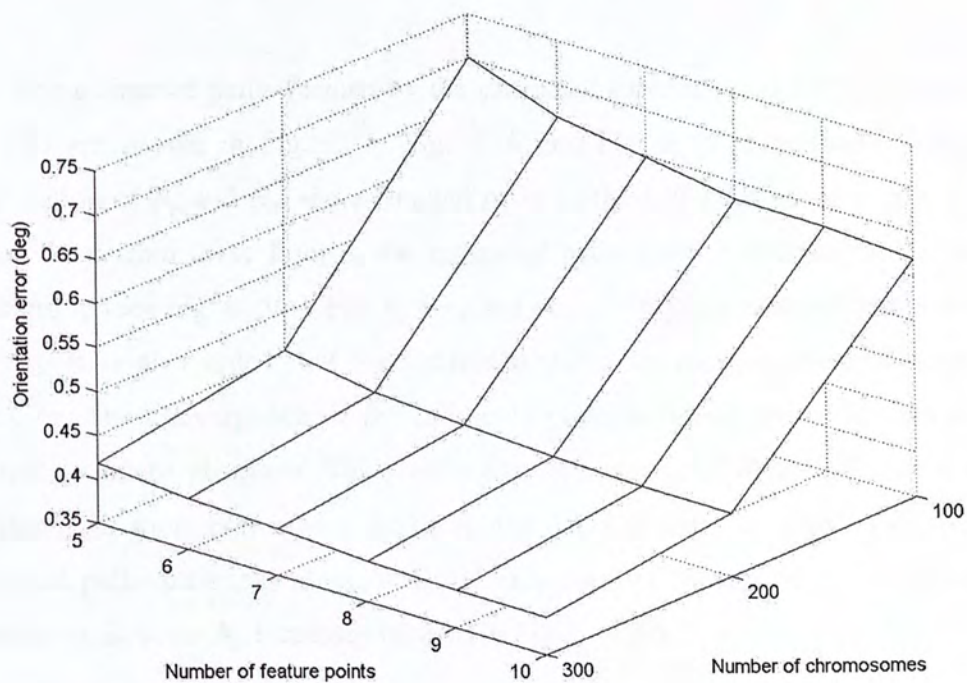
### 6.5.2 Results

The averaged results ( $E_p$ ,  $E_o$  and processing time) over the total number of processed frames in the 100 trials are shown in Fig. 6. 4. The  $E_p$  and  $E_o$  are varied from 3.4 cm to 4.3 cm and from  $0.4^\circ$  to  $0.7^\circ$  respectively. Seen from Fig. 6. 4(a) and Fig. 6. 4(b),  $E_p$  and  $E_o$  decrease along with the increments of population size  $N_p$  and number of feature points  $N_m$ . Since  $N_p$  describes the searching power of GA, the global optimum (i.e. actual robot state) is more probably to be found when  $N_p$  increases. Furthermore, as  $N_m$  increases, two influences to the corresponding fitness landscape occur: 1) the number of local optimum is reduced and 2) the differences between the global and local optimum are increased. As a result, the increment of  $N_m$  leads to reductions of  $E_p$  and  $E_o$ .

It is observed that there are some small oscillations in the values of  $E_p$  and  $E_o$  along the increments of  $N_p$  and  $N_m$ . A reason for these is that the increments of  $N_p$  and  $N_m$  cause a longer processing time per frame as shown in Fig. 6. 4(c). Hence, the time interval between the capture time of the current processing frame and the last processed frame increases. As the search space size depends on the maximum robot traveling distance in that time interval, the increments of  $N_p$  and  $N_m$  lead to increment of search space size. Therefore, effect of the search power enhancement due to the increases of  $N_p$  and  $N_m$  is compensated by the increase of the search space size.

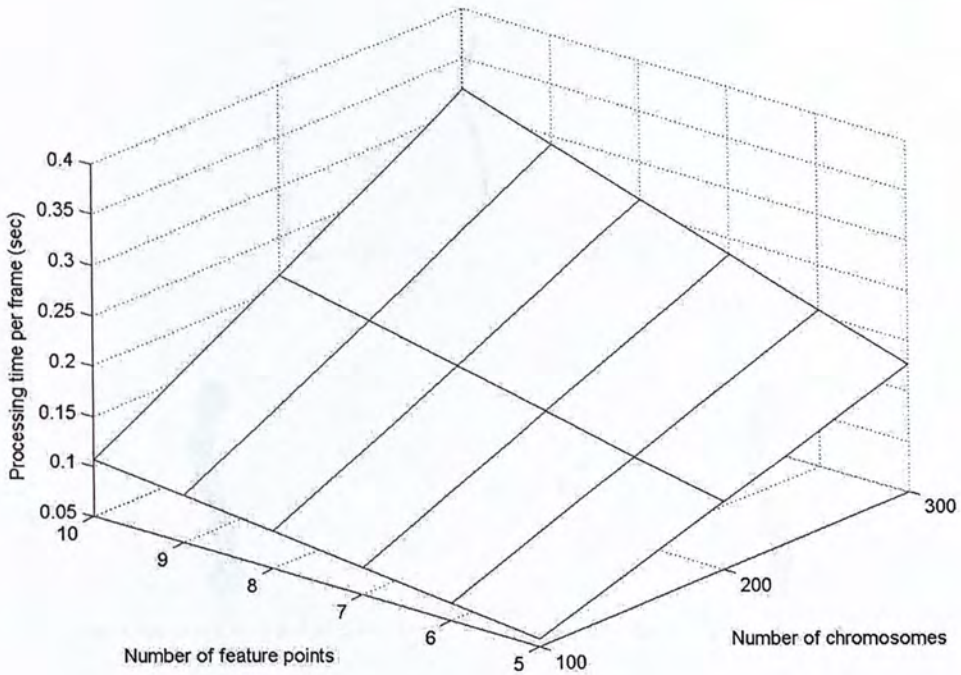


(a)



(b)

Fig. 6. 4 Experimental results of feature points with quantized noise: (a) average position errors, (b) average orientation errors and (c) average processing time per frame.



(c)

Fig. 6.4 (continued)

The estimated paths formed by the estimated camera states for  $N_p = 100, 200$  and  $300$  are shown in Fig. 6.5, Fig. 6.6 and Fig. 6.7 respectively. For each combination of  $N_p$  and  $N_m$ , the estimated paths in the 100 trials are shown in a sub-figure. Seen from these figures, the estimated paths are concentrated at the actual paths group (see Fig. 6.5(a), Fig. 6.6(a) and Fig. 6.7(a)) for all combinations of  $N_p$  and  $N_m$ . It is also noted that the estimated paths are more convergent when  $N_p$  increases. The convergence of the estimated paths indicates the accuracies of the estimated camera positions. The smaller the average error of the estimate camera positions, the more convergent the estimated paths. Hence, the convergence of the estimated paths increases along with the increment of  $N_p$  can be explained by the decrease in  $E_p$  when  $N_p$  increases (shown in Fig. 6.4(a)).

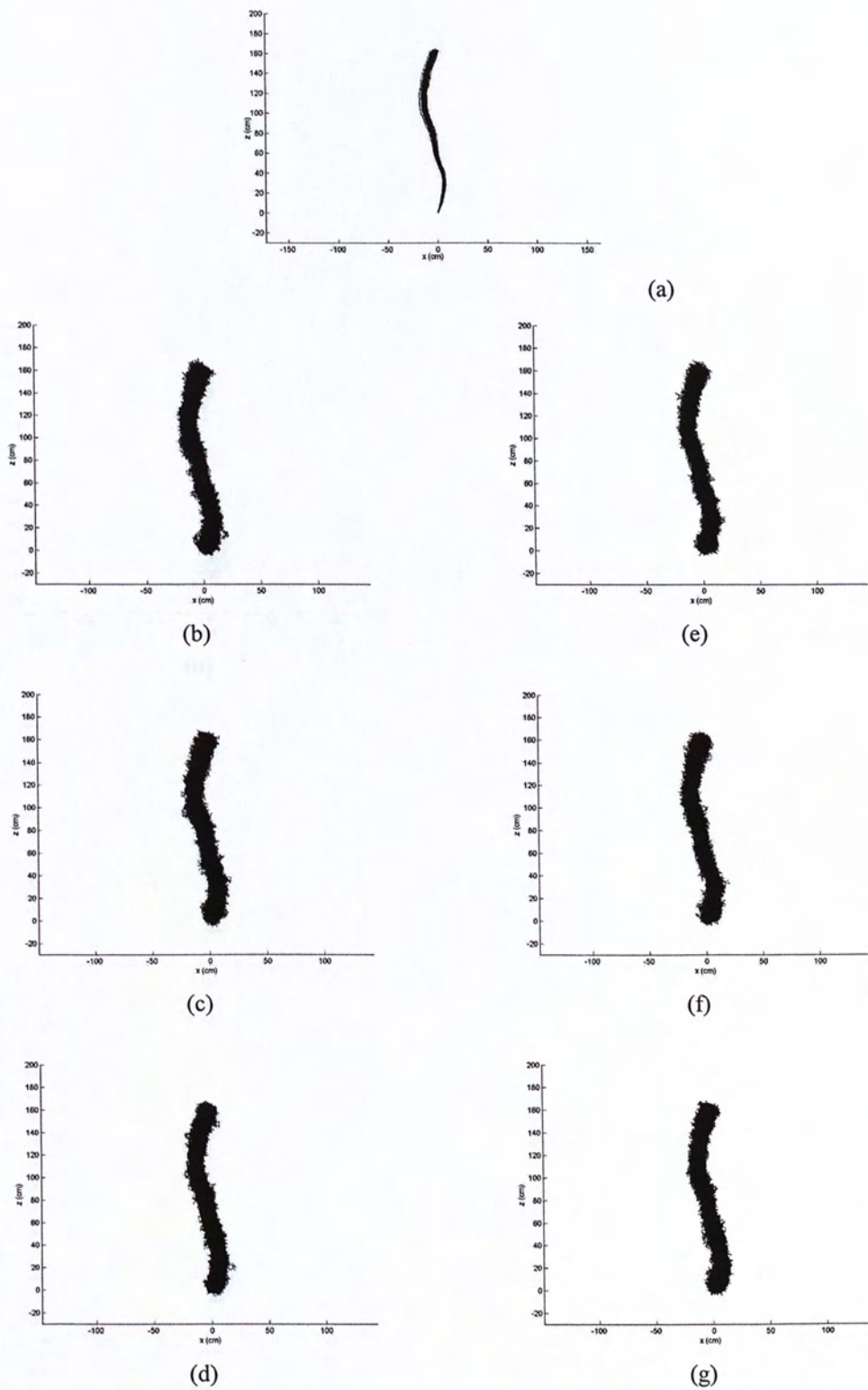


Fig. 6. 5 (a) Actual paths and (b-g) estimated paths of feature points with quantized noise,  $N_p=100$  and (b)  $N_m=5$ , (c)  $N_m=6$ , (d)  $N_m=7$ , (e)  $N_m=8$ , (f)  $N_m=9$  and (g)  $N_m=10$ .

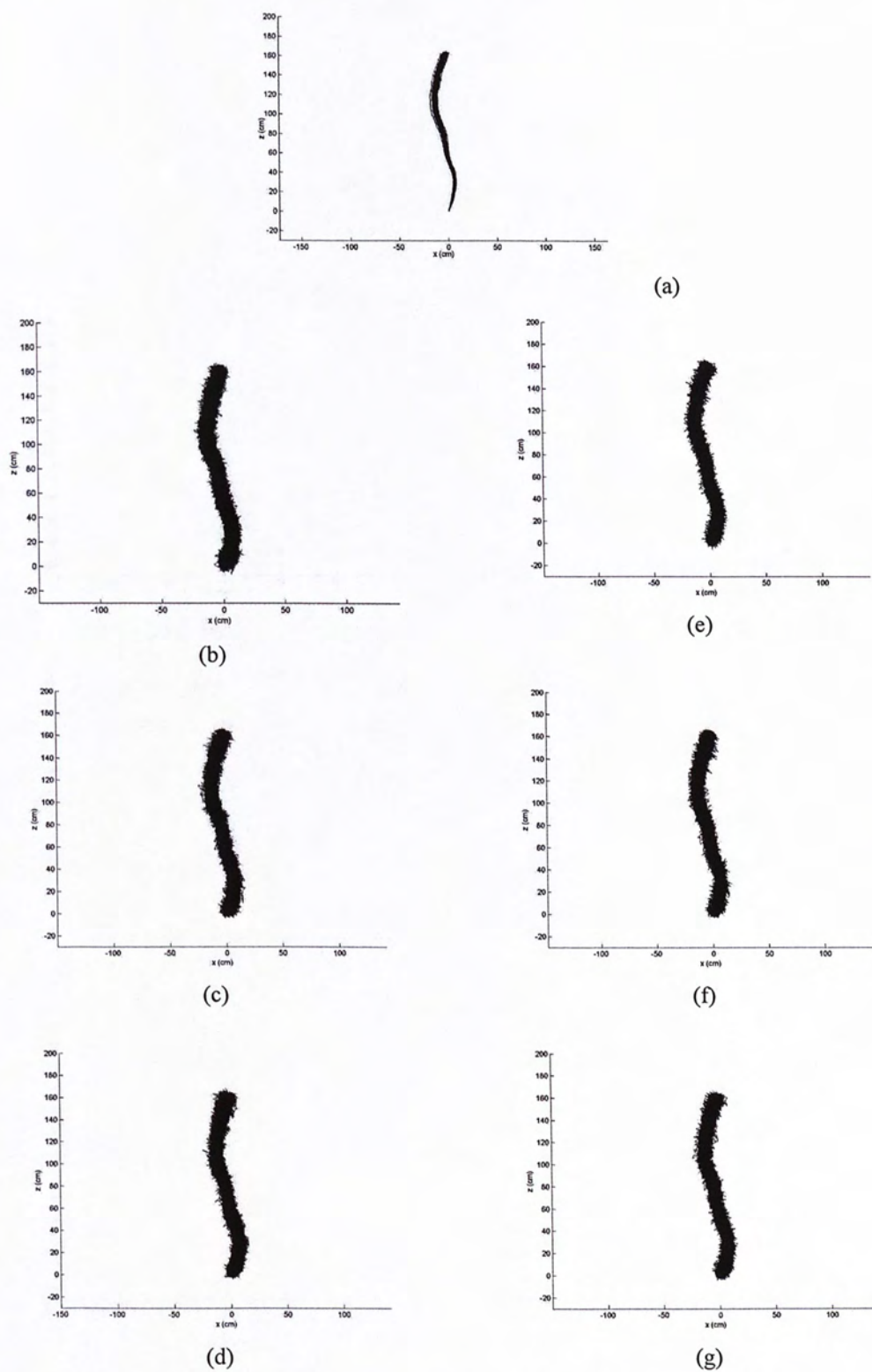


Fig. 6. 6 (a) Actual paths and (b-g) estimated paths of feature points with quantized noise,  $N_p=200$  and (b)  $N_m=5$ , (c)  $N_m=6$ , (d)  $N_m=7$ , (e)  $N_m=8$ , (f)  $N_m=9$  and (g)  $N_m=10$ .



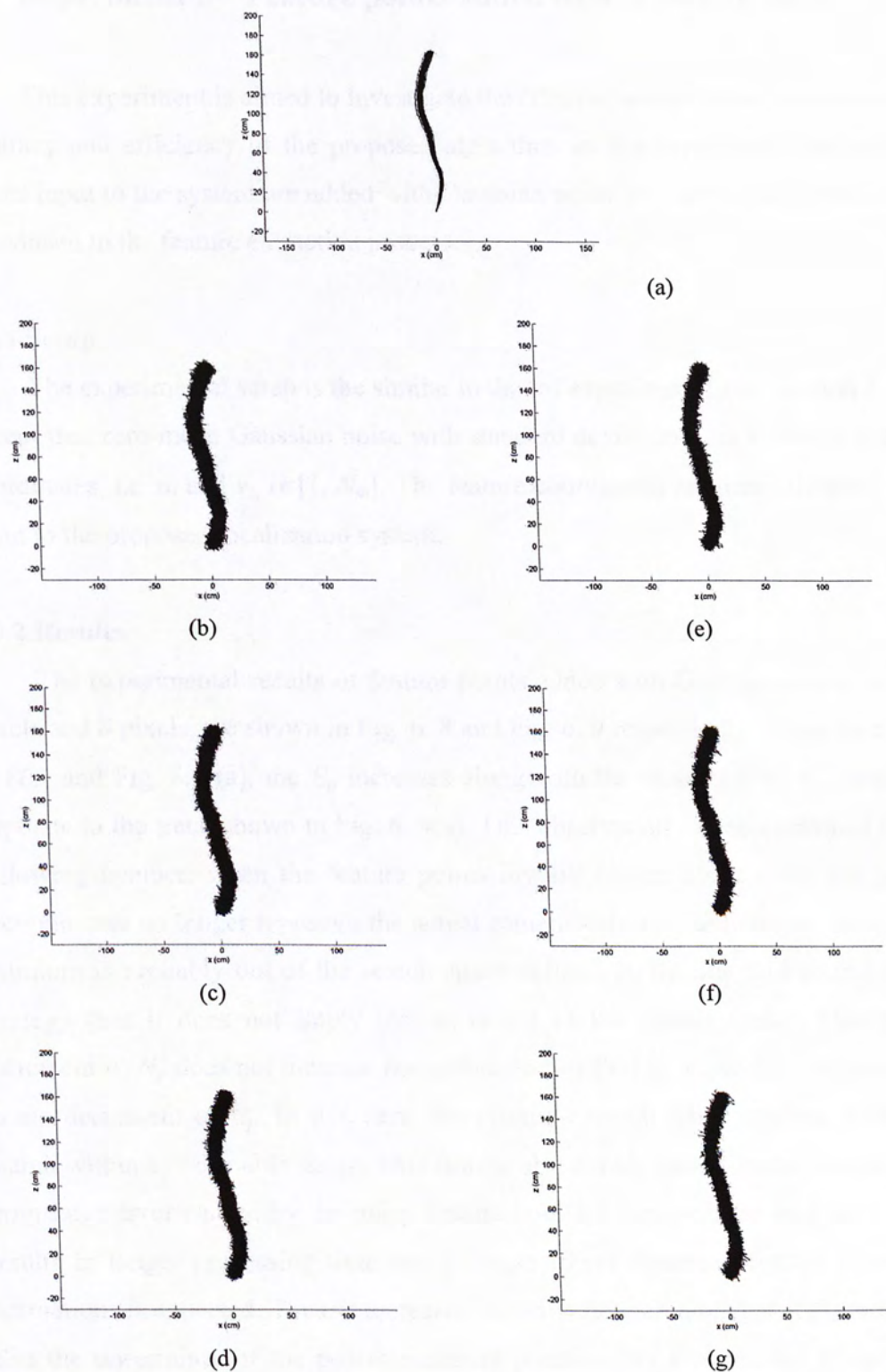


Fig. 6.7 (a) Actual paths and (b-g) estimated paths of feature points with quantized noise,  $N_p=300$  and (b)  $N_m=5$ , (c)  $N_m=6$ , (d)  $N_m=7$ , (e)  $N_m=8$ , (f)  $N_m=9$  and (g)  $N_m=10$ .

## 6.6 Experiment 2 – Feature points added with Gaussian noise

This experiment is aimed to investigate the effect of noisy feature points on the accuracy and efficiency of the proposed algorithm. In the experiment, the feature points input to the system are added with Gaussian noise in order to model the error introduced in the feature extraction process.

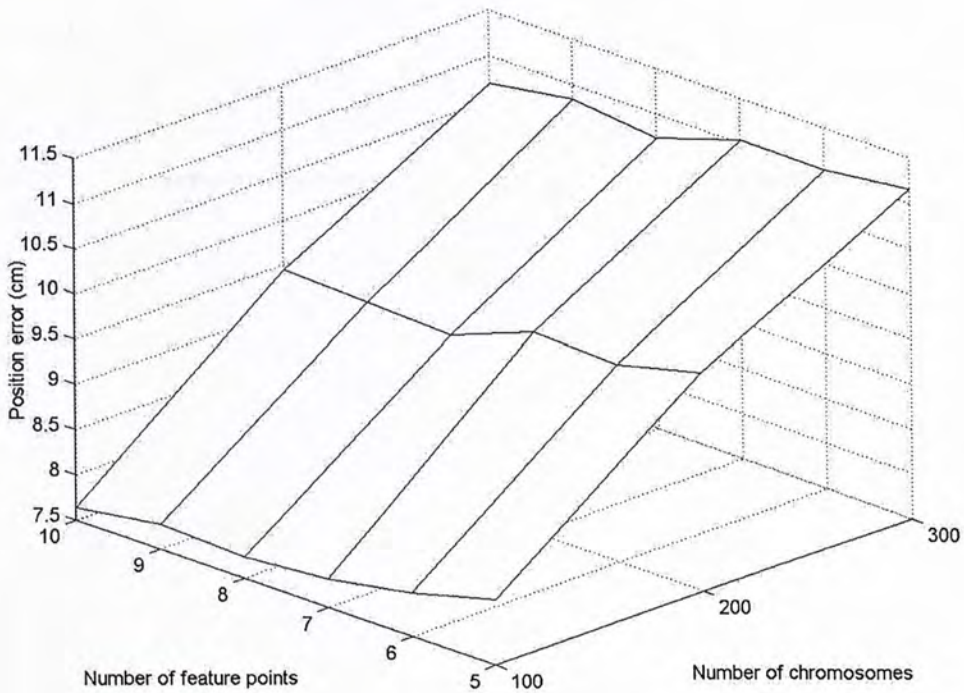
### 6.6.1 Setup

The experimental setup is the similar to that of experiment 1 (see section 6.5.1) except that zero-mean Gaussian noise with standard deviation  $\sigma_n$  is added to feature coordinates, i.e.  $u_i$  and  $v_i$ ,  $i \in [1, N_m]$ . The feature coordinates are then quantized and input to the proposed localization system.

### 6.6.2 Results

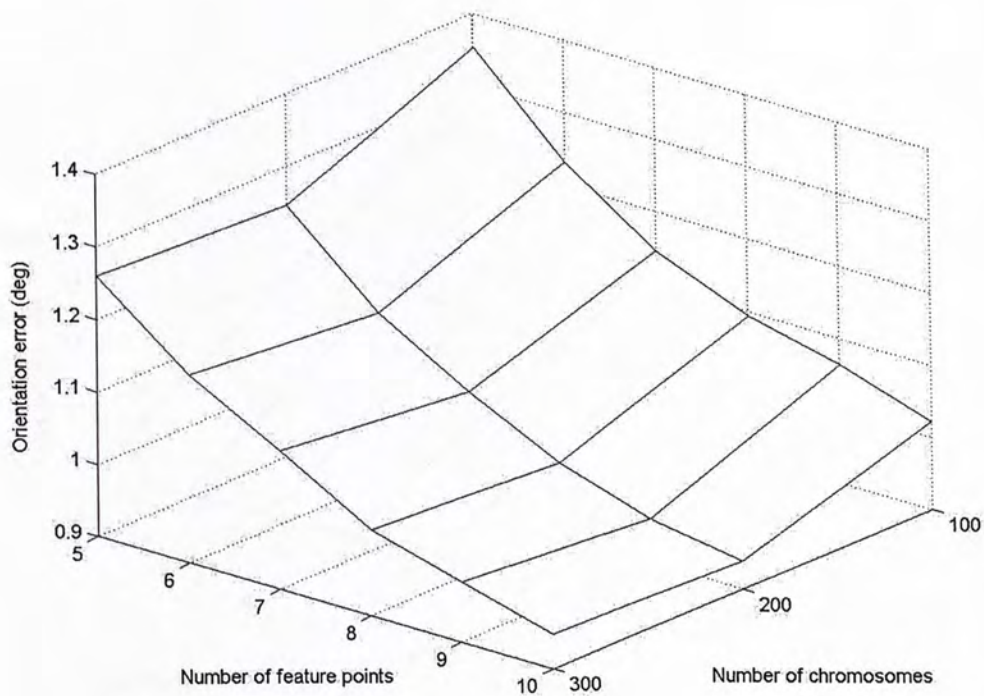
The experimental results of feature points added with Gaussian noise,  $\sigma_n = 4$  pixels and 8 pixels, are shown in Fig. 6. 8 and Fig. 6. 9 respectively. Seen from Fig. 6. 8(a) and Fig. 6. 9(a), the  $E_p$  increases along with the increment of  $N_p$ , which is opposite to the trend shown in Fig. 6. 4(a). This observation can be explained in the following manner: when the feature points involve higher noise level, the global optimum can no longer represent the actual camera state  $\mathbf{r}_0$ . Furthermore, the global optimum is probably out of the search space defined by the adaptive search space strategy (but it does not imply that  $\mathbf{r}_0$  is out of the search space). Hence, the increment of  $N_p$  does not increase the probability of finding  $\mathbf{r}_0$ , which does not lead to the decrement of  $E_p$ . In this case, the adaptive search space strategy limits the search within a reasonable range. This feature also avoids the estimated camera state from large error caused by the noisy feature points. Moreover, the increment of  $N_p$  results in longer processing time that a longer travel distance between successive estimations is expected. Thus, it increases not only the search ranges of  $P_x$  and  $P_z$  but also the uncertainty of the possible camera position. As a result, the  $E_p$  increases along with the increment of  $N_p$  for noisy feature points. On the other hand, as the increment of feature points that involve noise does not provide addition information to the search, the effect of  $N_m$  on the  $E_p$  is small.

Similar to the case of  $E_p$ , the increment of  $N_p$  does not lead to the decrement of  $E_o$  as 1) the global optimum may not equal to  $\mathbf{r}_0$  and 2)  $\mathbf{r}_0$  may not be included in the search space. On the other hand, the range of the orientation genes  $\Omega_o$  is fixed that the increment of  $N_p$  does not lead to the increment of  $E_o$ . Fig. 6. 8(b) and Fig. 6. 9(b) empirically show the small effect of  $N_m$  to the  $E_o$ . Seen from Fig. 6. 4(c), Fig. 6. 8(c) and Fig. 6. 9(c), the processing time per frame does not depend on the noise level of feature points, which is only depends on the  $N_m$  and the  $N_p$ .

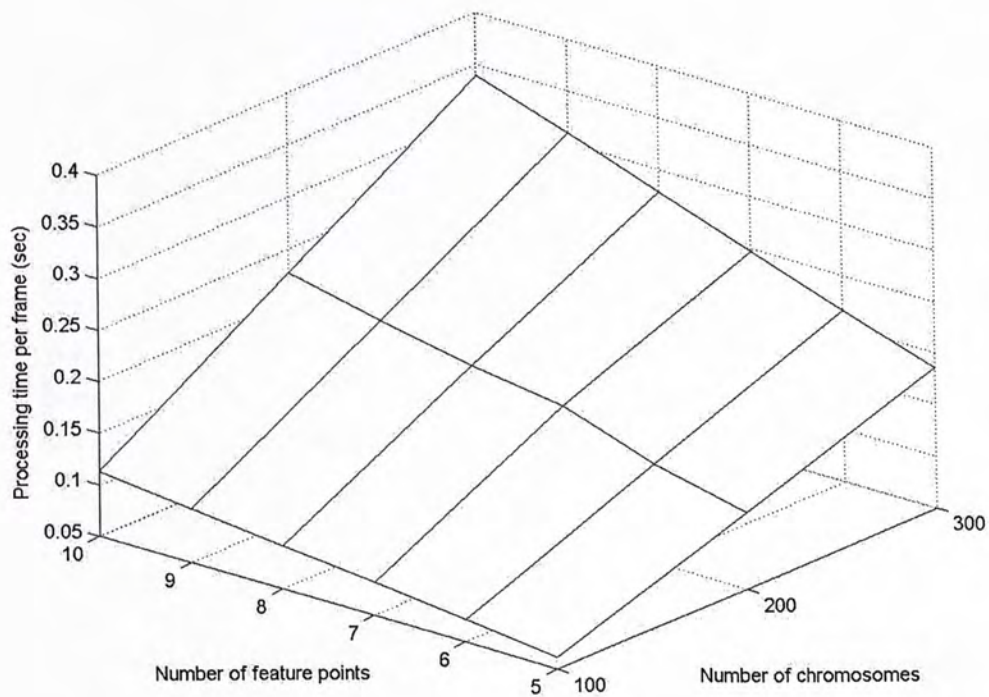


(a)

Fig. 6. 8 Experimental results of feature points added with Gaussian noise ( $\sigma_n = 4$  pixels) : (a) average position errors, (b) average orientation errors and (c) average processing time per frame.

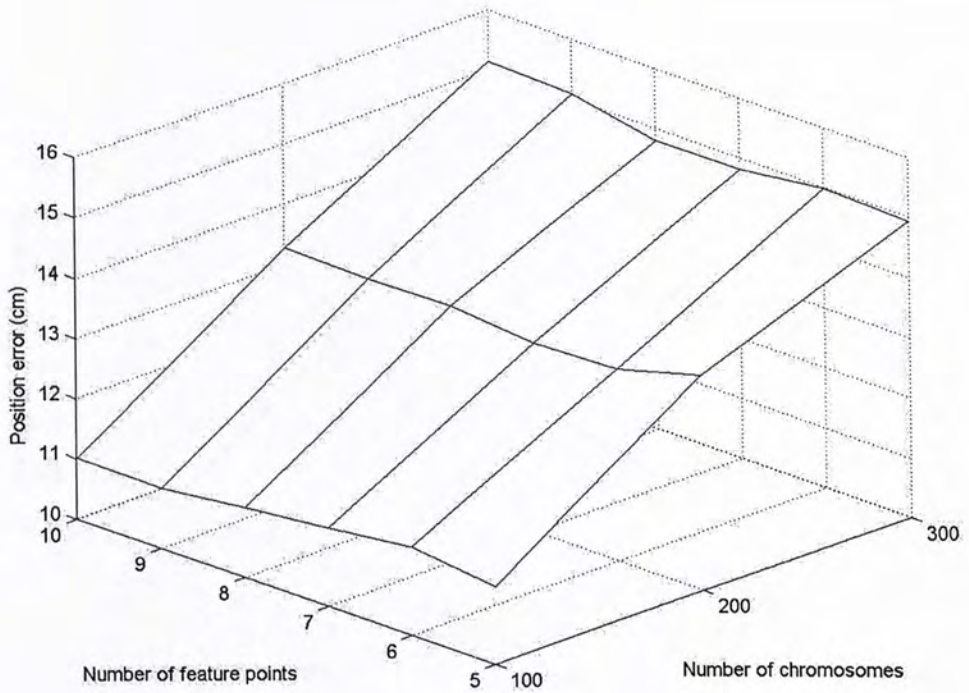


(b)

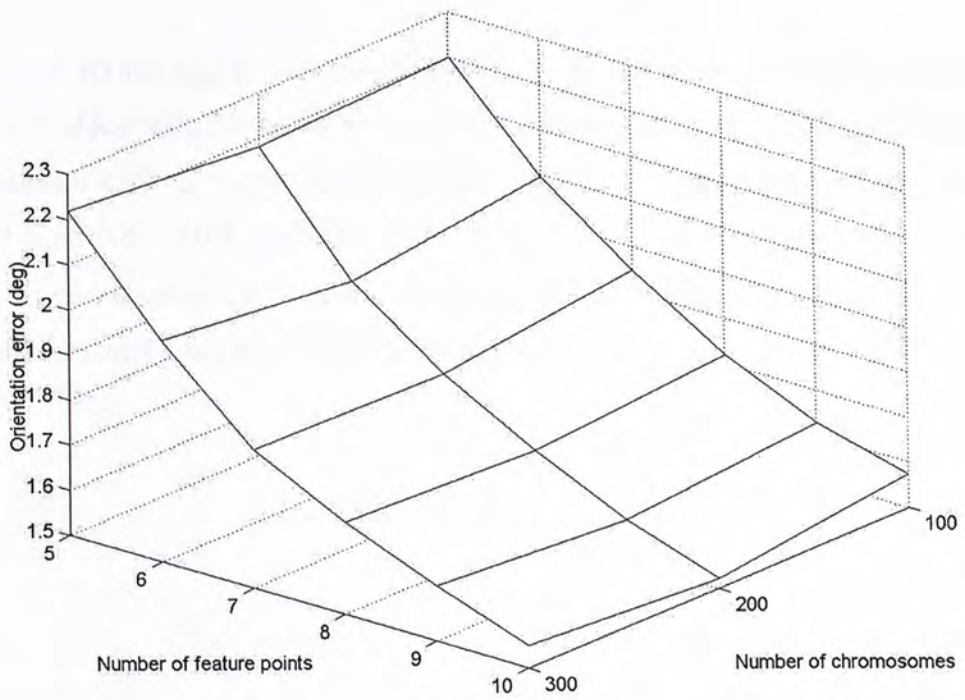


(c)

Fig. 6. 8 (continued)

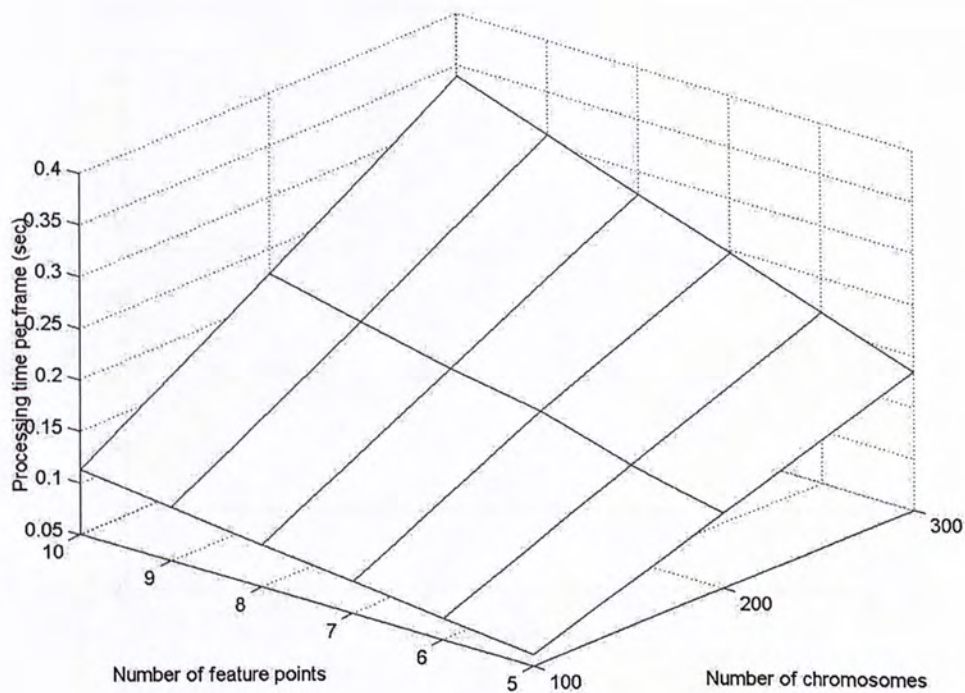


(a)



(b)

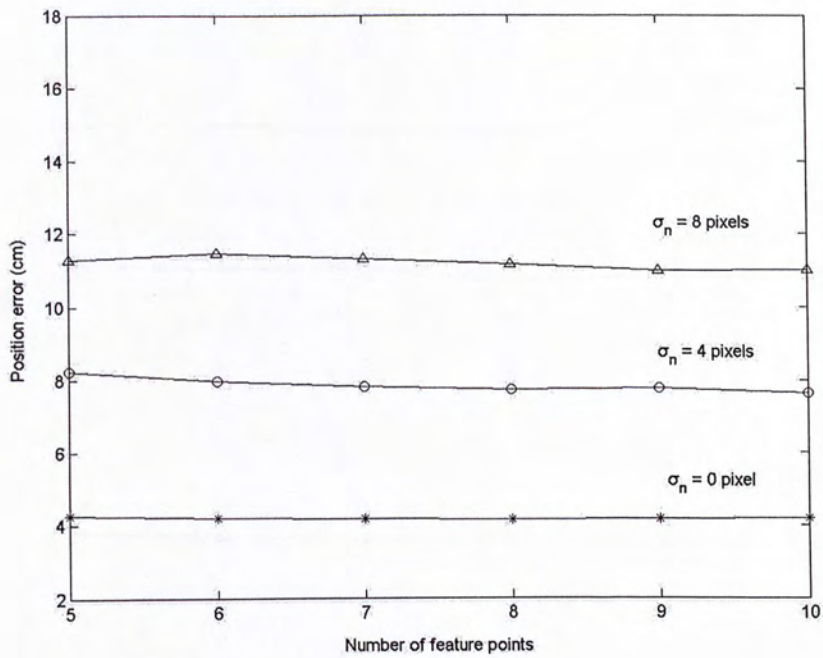
Fig. 6.9 Experimental results of feature points added with Gaussian noise ( $\sigma_n = 8$  pixels) : (a) average position errors, (b) average orientation errors and (c) average processing time per frame.



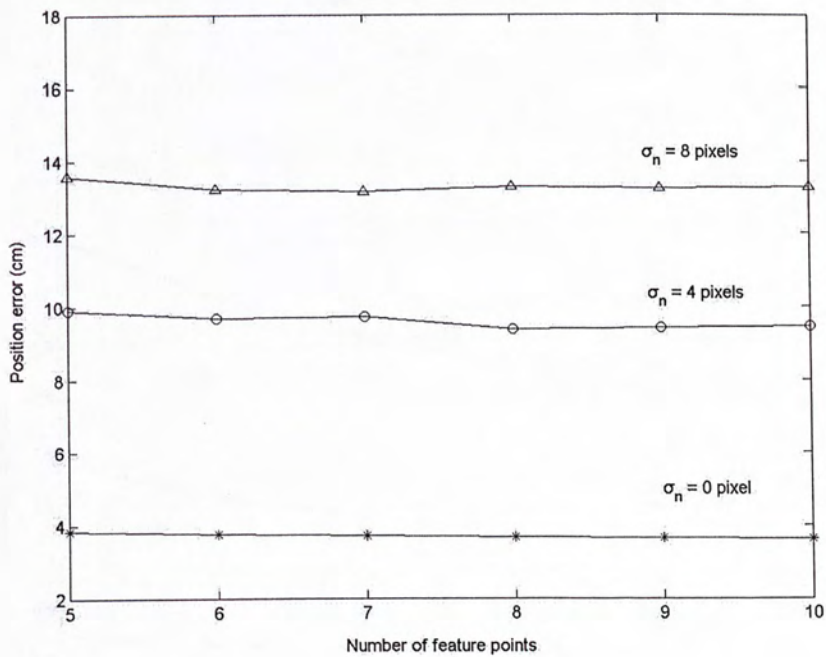
(c)

Fig. 6.9 (continued)

Fig. 6.10 and Fig. 6.11 show the  $E_p$  and the  $E_o$  respectively where the feature points are added with Gaussian noise with  $\sigma_n$  varied from 0 to 8 pixels. Compare with the results for  $\sigma_n = 0$ , the increments of  $E_p$  and  $E_o$  are ranged from 3.4 cm to 7.5 cm and from  $0.4^\circ$  to  $0.8^\circ$  respectively for  $\sigma_n = 4$ . The increments of  $E_p$  and  $E_o$  are ranged from 6.8 cm to 11.7 cm and from  $1.0^\circ$  to  $1.8^\circ$  respectively for  $\sigma_n = 8$ . It is observed that the  $E_p$  and the  $E_o$  increase along with  $\sigma_n$ .

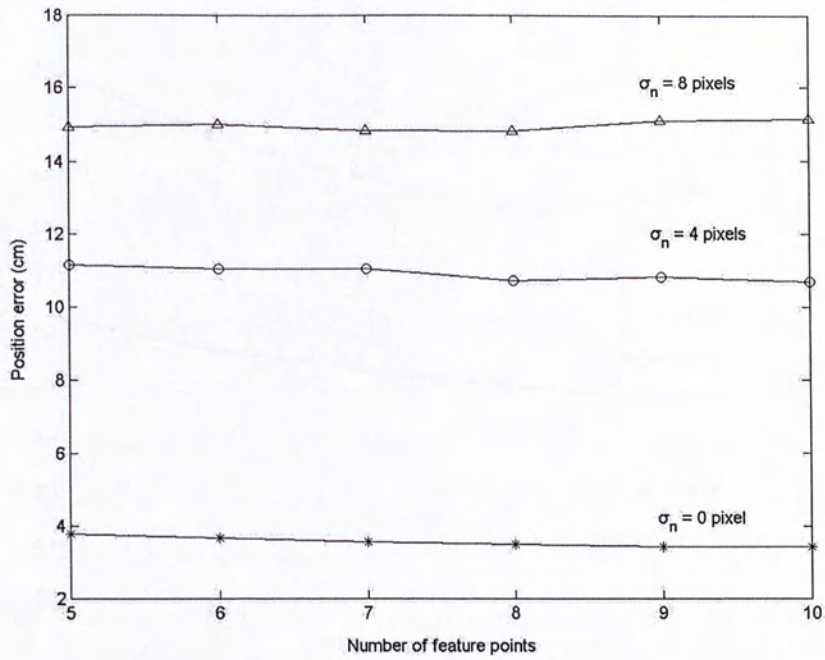


(a)



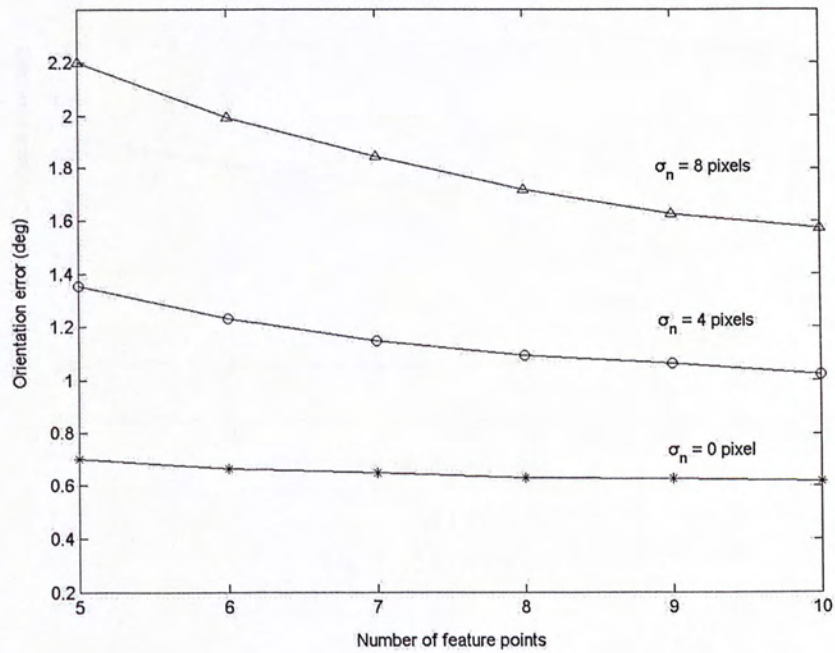
(b)

Fig. 6. 10 Position errors for the population size (a) 100, (b) 200 and (c) 300. In each sub-figure, the standard deviation  $\sigma_n$  of the added Gaussian noise is varied from 0 to 8 pixels.



(c)

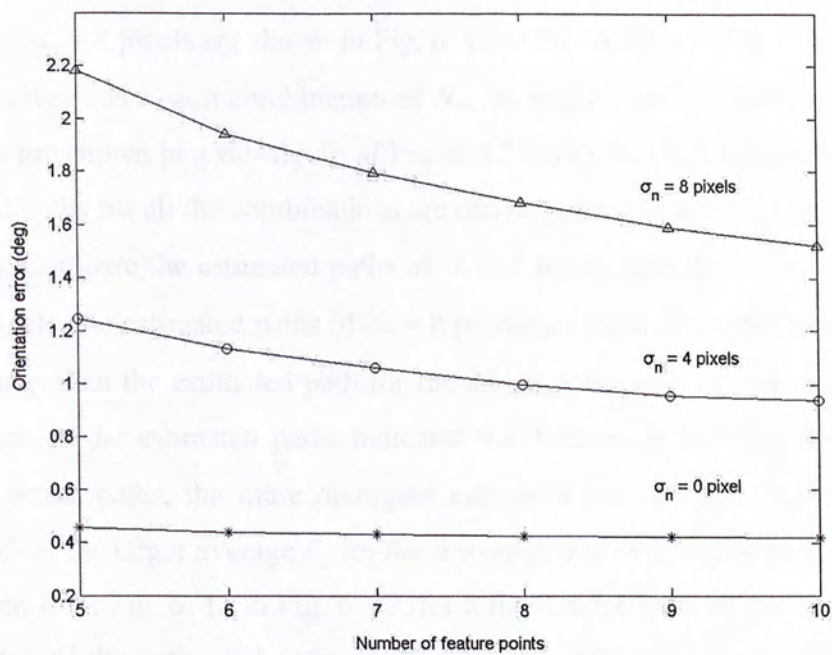
Fig. 6.9 (continued)



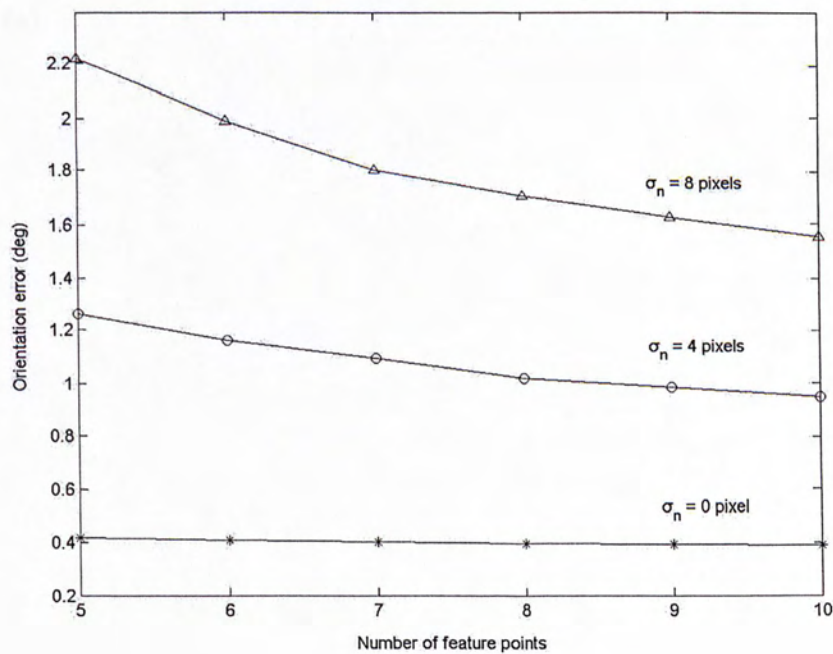
(a)

Fig. 6.11 Orientation errors for the population size (a) 100, (b) 200 and (c) 300. In each sub-figure, the standard deviation  $\sigma_n$  of the added Gaussian noise is varied from 0 to 8 pixels.





(b)



(c)

Fig. 6.11 (continued)

---

The estimated paths of the feature points added with Gaussian noise:  $\sigma_n = 4$  pixels and  $\sigma_n = 8$  pixels are shown in Fig. 6. 12 to Fig. 6. 14 and Fig. 6. 15 to Fig. 6. 17 respectively. For each combination of  $N_m$ ,  $N_p$  and  $\sigma_n$ , the estimated paths in the 100 trials are shown in a sub-figure of Fig. 6. 12 to Fig. 6. 17. It is observed that the estimated paths for all the combinations are converged to the actual paths group (Fig. 6. 12(a)). Compare the estimated paths of  $\sigma_n = 4$  pixels with the estimated paths of  $\sigma_n = 8$  pixels, the estimated paths of  $\sigma_n = 8$  pixels are more diverged from the actual paths group than the estimated path for the added noise with  $\sigma_n = 4$  pixels. As the divergence of the estimated paths indicates the differences between the estimated and the actual paths, the more divergent estimated path of  $\sigma_n = 8$  pixels can be explained by the larger average  $E_p$  for the feature points with higher noise level.

Seen from Fig. 6. 12 to Fig. 6. 17, for a fixed noise level of feature points, the divergence of the estimated paths increases along with the increment of  $N_p$ . It is because the  $E_p$  increases along with the increment of  $N_p$  as shown in Fig. 6. 8 (a) and Fig. 6. 9 (a).

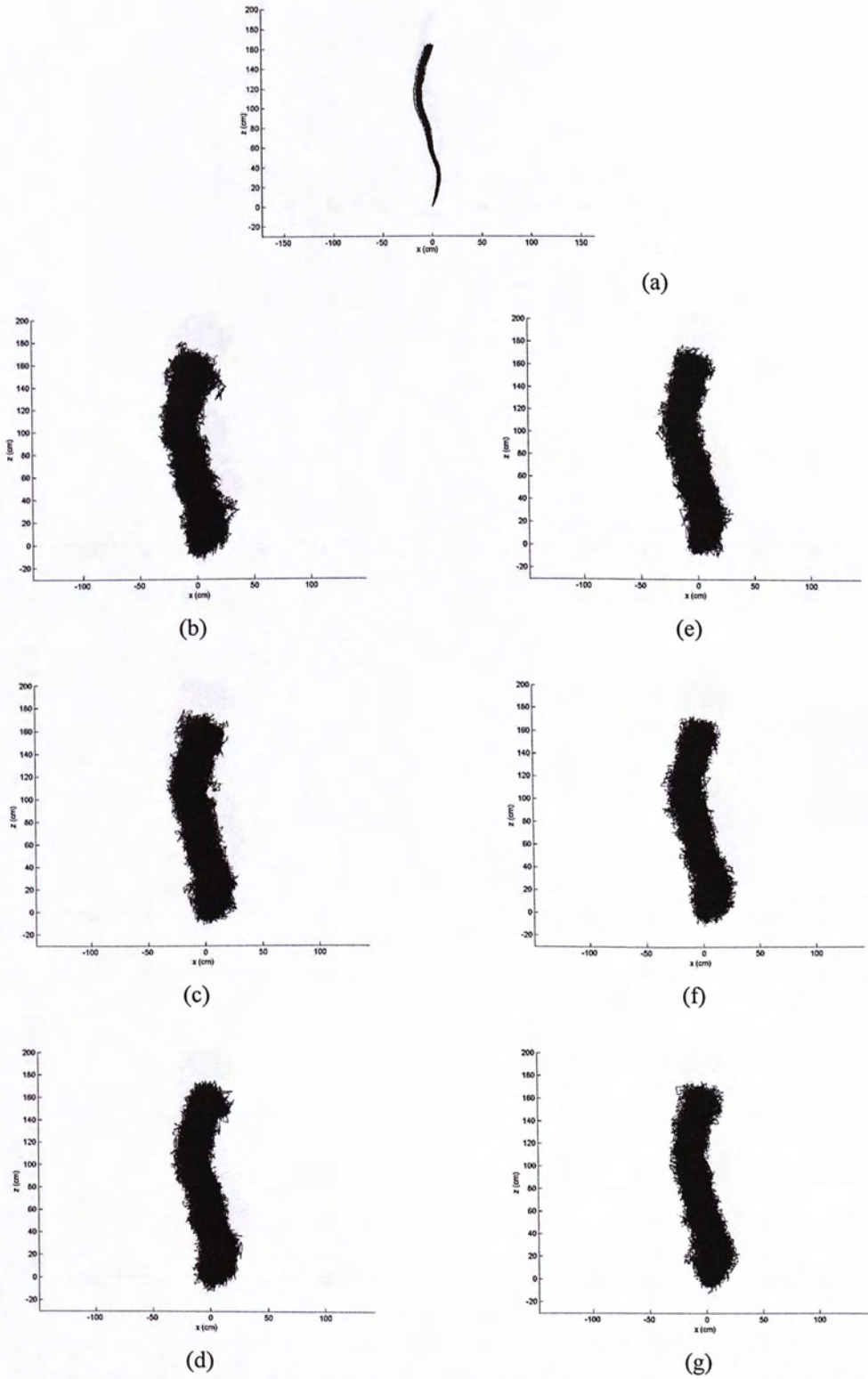


Fig. 6.12 (a) Actual paths and (b-g) estimated paths of feature points added with Gaussian noise ( $\sigma_n = 4$  pixels),  $N_p=100$  and (b)  $N_m=5$ , (c)  $N_m=6$ , (d)  $N_m=7$ , (e)  $N_m=8$ , (f)  $N_m=9$  and (g)  $N_m=10$ .

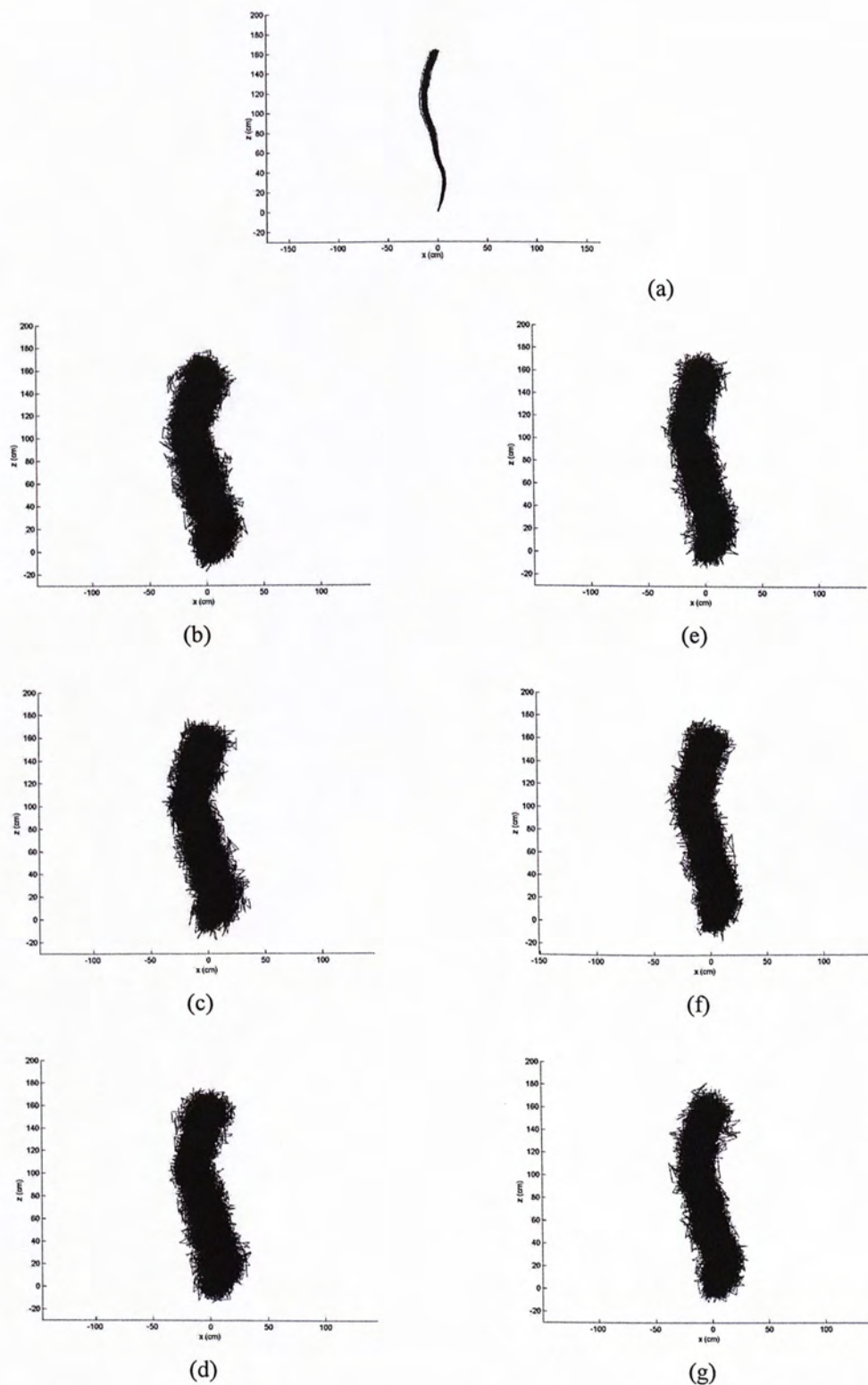


Fig. 6.13 (a) Actual paths and (b-g) estimated paths of feature points added with Gaussian noise ( $\sigma_n = 4$  pixels),  $N_p=200$  and (b)  $N_m=5$ , (c)  $N_m=6$ , (d)  $N_m=7$ , (e)  $N_m=8$ , (f)  $N_m=9$  and (g)  $N_m=10$ .

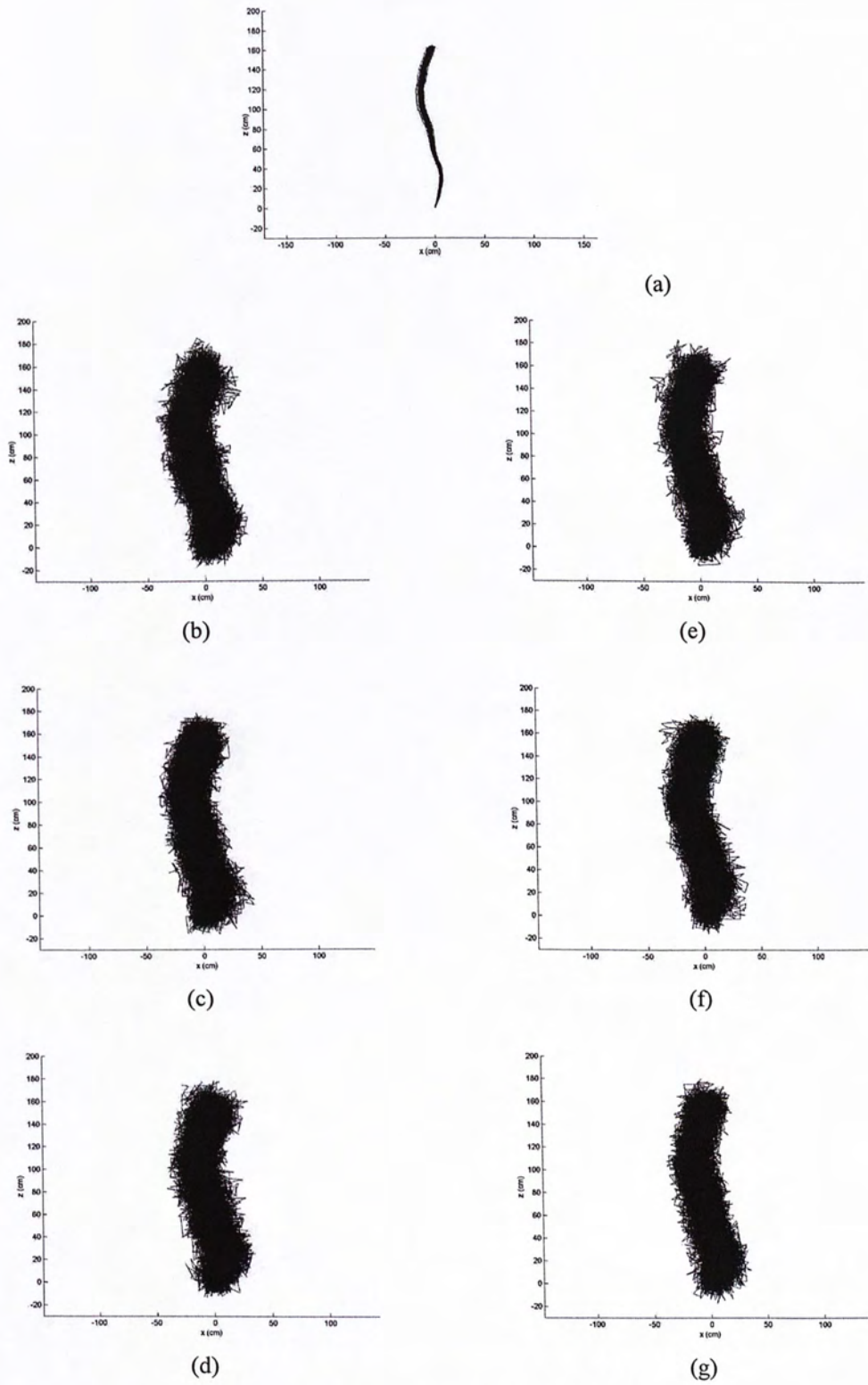


Fig. 6.14 (a) Actual paths and (b-g) estimated paths of feature points added with Gaussian noise ( $\sigma_n = 4$  pixels),  $N_p=300$  and (b)  $N_m=5$ , (c)  $N_m=6$ , (d)  $N_m=7$ , (e)  $N_m=8$ , (f)  $N_m=9$  and (g)  $N_m=10$ .

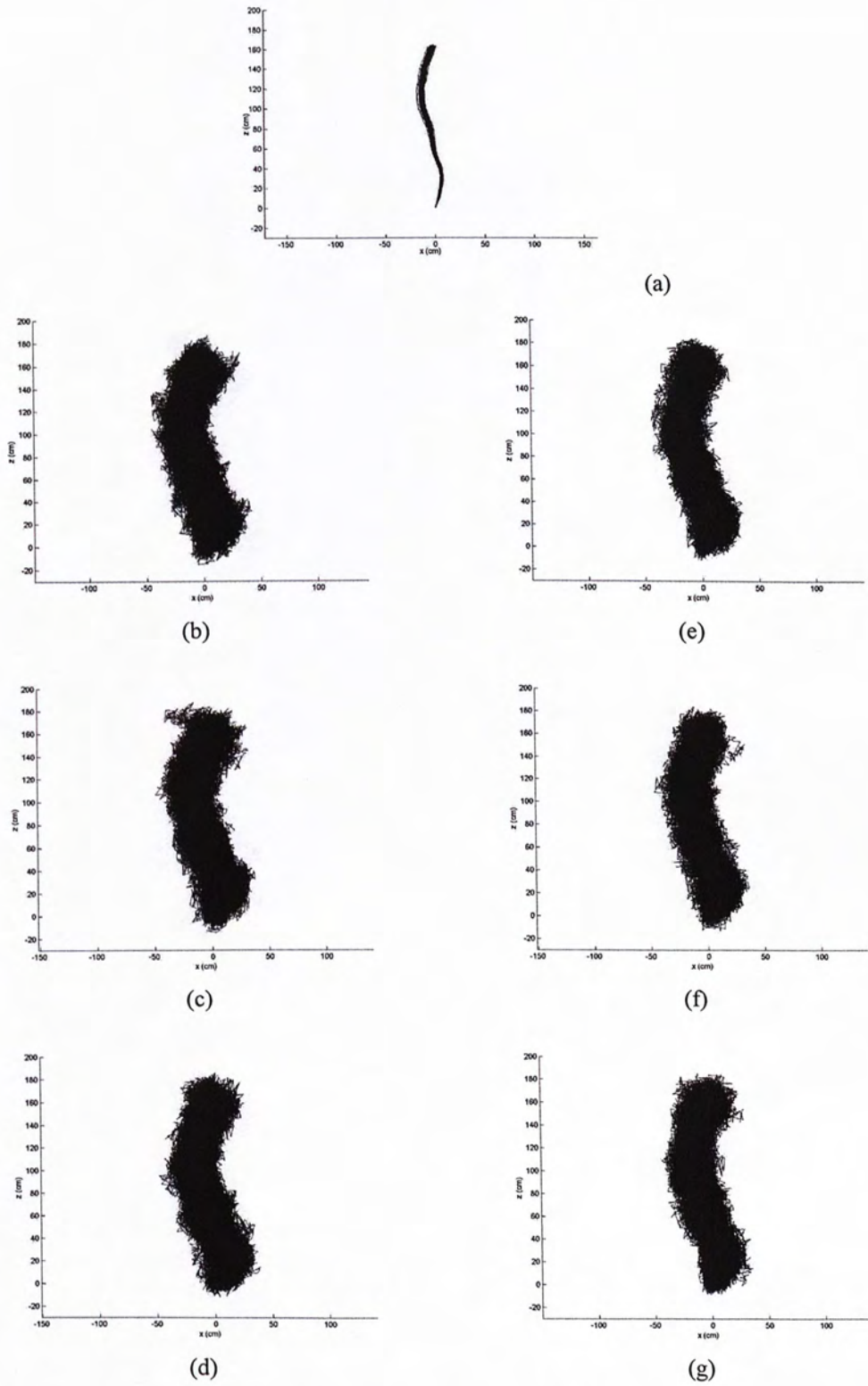


Fig. 6.15 (a) Actual paths and (b-g) estimated paths of feature points added with Gaussian noise ( $\sigma_n = 8$  pixels),  $N_p=100$  and (b)  $N_m=5$ , (c)  $N_m=6$ , (d)  $N_m=7$ , (e)  $N_m=8$ , (f)  $N_m=9$  and (g)  $N_m=10$ .

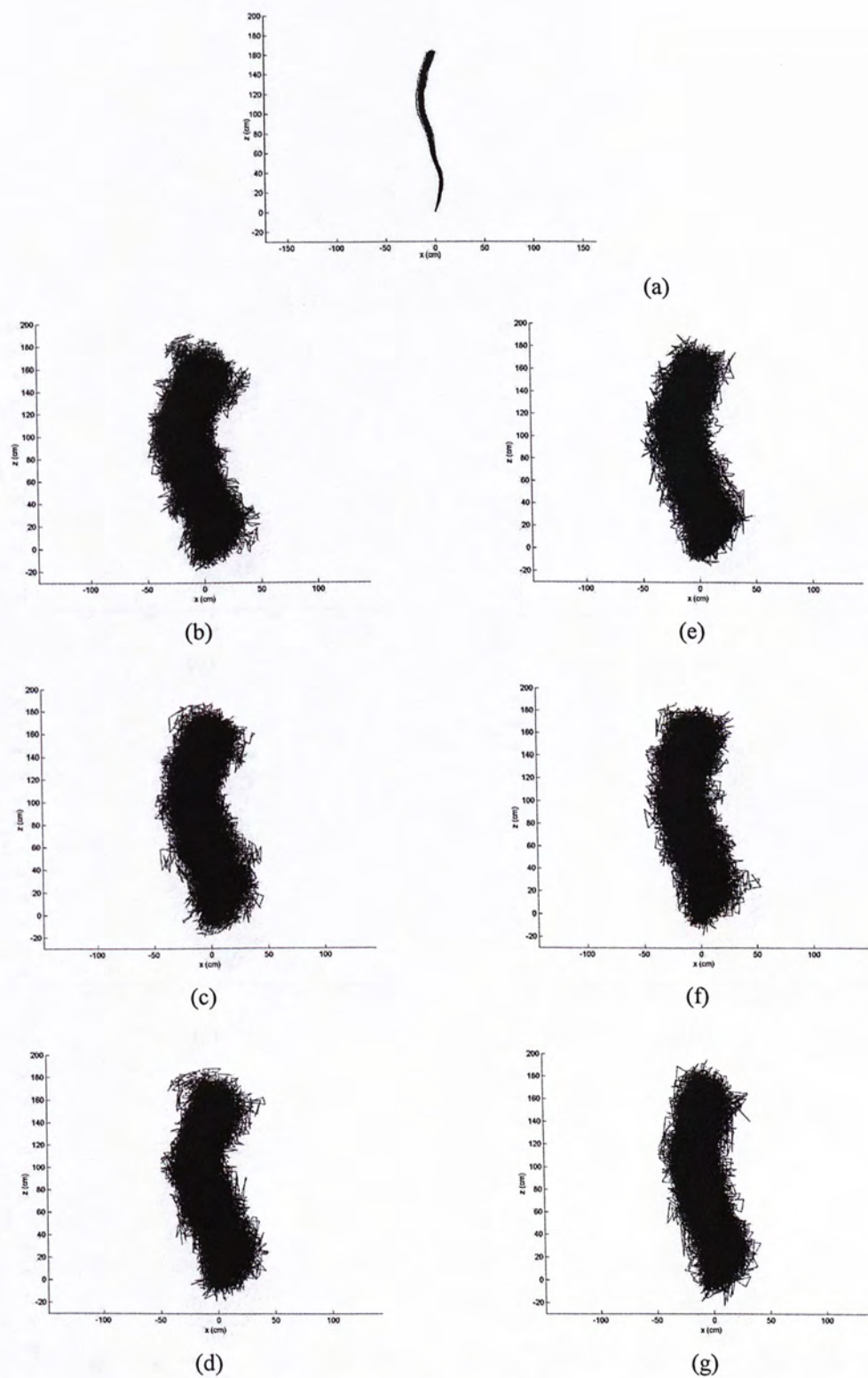


Fig. 6.16 (a) Actual paths and (b-g) estimated paths of feature points added with Gaussian noise ( $\sigma_n = 8$  pixels),  $N_p=200$  and (b)  $N_m=5$ , (c)  $N_m=6$ , (d)  $N_m=7$ , (e)  $N_m=8$ , (f)  $N_m=9$  and (g)  $N_m=10$ .

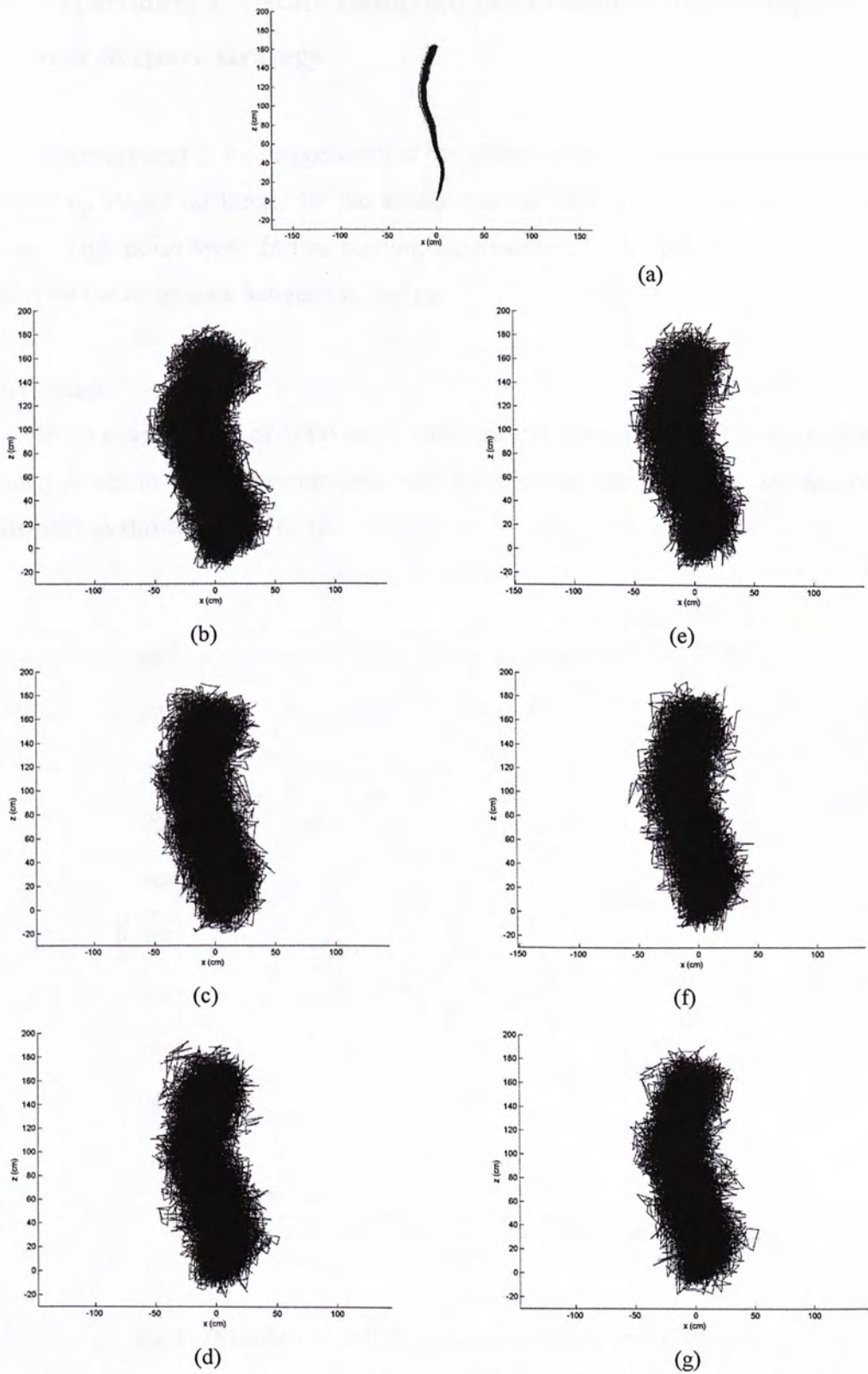


Fig. 6.17 (a) Actual paths and (b-g) estimated paths of feature points added with Gaussian noise ( $\sigma_n = 8$  pixels),  $N_p=300$  and (b)  $N_m=5$ , (c)  $N_m=6$ , (d)  $N_m=7$ , (e)  $N_m=8$ , (f)  $N_m=9$  and (g)  $N_m=10$ .



## 6.7 Experiment 3 – Noise reduction performance of the adaptive search space strategy

In experiment 2, we suggested that the global optimum in the landscape of  $f(\mathbf{r})$ , namely  $\mathbf{r}_g$ , might no longer be the actual camera state  $\mathbf{r}_0$  when the feature points involve high noise level. In this section, we demonstrate the effect of noisy feature points on the difference between  $\mathbf{r}_g$  and  $\mathbf{r}_0$ .

### 6.7.1 Setup

In an environment of  $1000 \text{ cm} \times 1000 \text{ cm}$  that consists of 100 landmarks with known positions at the surrounding, 100 independent camera states are randomly generated as shown in Fig. 6. 18.

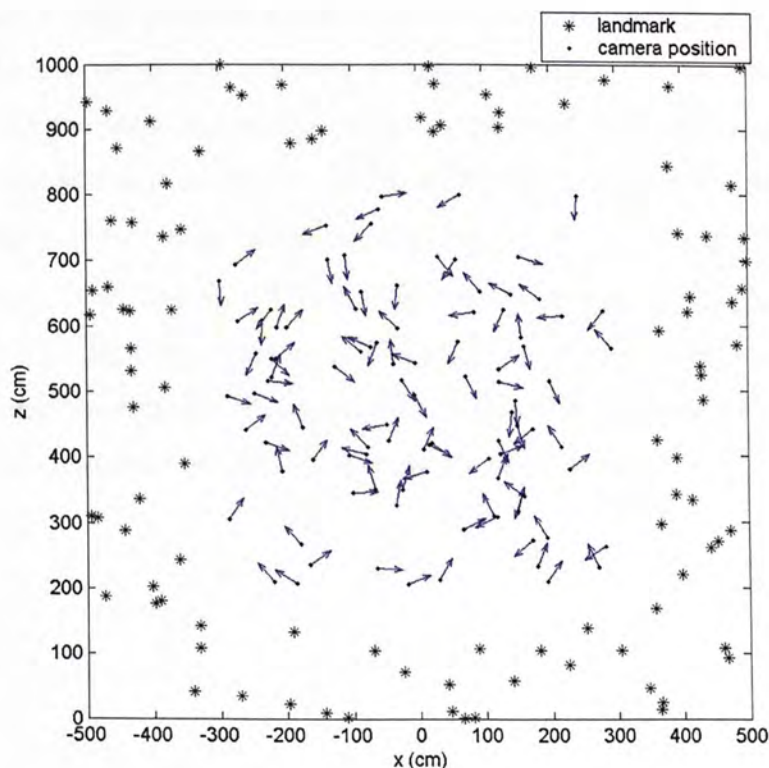


Fig. 6. 18 Landmarks and 100 independent camera states (Topview).

These camera states are regarded as the actual camera states, and each of which is in the form of  $[p_x, p_y, p_z, \theta_x, \theta_y, \theta_z]$  where  $[p_x, p_y, p_z]$  is the camera position vector represented in the Cartesian coordinates and  $[\theta_x, \theta_y, \theta_z]$  is the camera orientation vector represented in Euler angles.  $p_x, p_z$  and  $\theta_y$  are randomly generated within the ranges  $[-300 \text{ cm}, 300 \text{ cm}]$ ,  $[200 \text{ cm}, 800 \text{ cm}]$  and  $[-180^\circ, 180^\circ]$  respectively under uniform distribution. In Fig. 6. 18, the dots indicate the camera positions on the  $xz$ -plane (i.e.  $[p_x, p_z]$ ) and the arrows show the camera orientation (i.e.  $\theta_y$ ). The value changes of  $p_y, \theta_x$  and  $\theta_z$  caused by the oscillated walking motion are generated by the simulator as described in section 6.2.2. For each camera state, the corresponding input feature coordinates are generated by the simulator (see section 6.2.3). In order to investigate the effect of noisy feature points on the difference between  $\mathbf{r}_g$  and  $\mathbf{r}_0$ , the feature points are added with different levels of Gaussian noise and then quantized. The  $\mathbf{r}_g$  corresponding to each  $\mathbf{r}_0$  is obtained by the GA with a large population size  $N_p = 400$  and the setting is similar to that described in chapter 5, except that 1) the population size is fixed for every generation, 2) the search is terminated only when the number of generation reaches 300 and 3) the search is performed within a fixed search range set, which includes all the possible camera states in the environment: i.e.  $P_x \in [-500 \text{ cm}, 500 \text{ cm}]$ ,  $P_y \in [(p_{y0} - 3) \text{ cm}, (p_{y0} + 3) \text{ cm}]$ ,  $P_z \in [0 \text{ cm}, 1000 \text{ cm}]$ ,  $Q_\alpha \in [-\pi/2, \pi/2]$ ,  $Q_\beta \in [-\pi/2, \pi/2]$  and  $Q_l \in [-1, 1]$ , for every generation. Since GA is a stochastic method, each of the  $\mathbf{r}_g$  searches are repeated for 10 independent trials and the camera state corresponds to the smallest fitness among the 10 trials are regarded as  $\mathbf{r}_g$ .

### 6.7.2 Results

The experimental results of differences between  $\mathbf{r}_g$  and  $\mathbf{r}_0$  for the number of feature points  $N_m = 5$ , and the added Gaussian noise with standard deviation  $\sigma_n = 4$  pixels and 8 pixels are shown in Fig. 6. 19 and Fig. 6. 20 respectively. In these figures, the  $\mathbf{r}_0$  (dot) and its corresponding  $\mathbf{r}_g$  (cross) are linked by a dotted line. Hence, the lengths of the dotted lines represent the differences (on  $xz$ -plane) between  $\mathbf{r}_0$  and  $\mathbf{r}_g$  due to the noisy feature points. Seen from Fig. 6. 19 and Fig. 6. 20, the lengths of the dotted lines for  $\sigma_n = 8$  pixels are generally longer than those for  $\sigma_n = 4$  pixels. The 3D distance between  $\mathbf{r}_0$  and  $\mathbf{r}_g$  averaged over the 100 independent camera states, namely  $\Delta r$ , are 32.0 cm and 69.6 cm for  $\sigma_n = 4$  pixels and 8 pixels respectively. The sorted  $\Delta r$  for feature points with  $\sigma_n = 4$  pixels and 8 pixels are shown in Fig. 6. 21. The figure illustrates that the distance between  $\mathbf{r}_0$  and  $\mathbf{r}_g$  increases along with the increment of the noise level of the feature points. Hence, when the feature points involve high noise level, the global optimum can no longer be the actual camera state. As a result, localizing the camera by searching for the global optimum in a search space that includes all possible camera states in the environment might induce large errors.

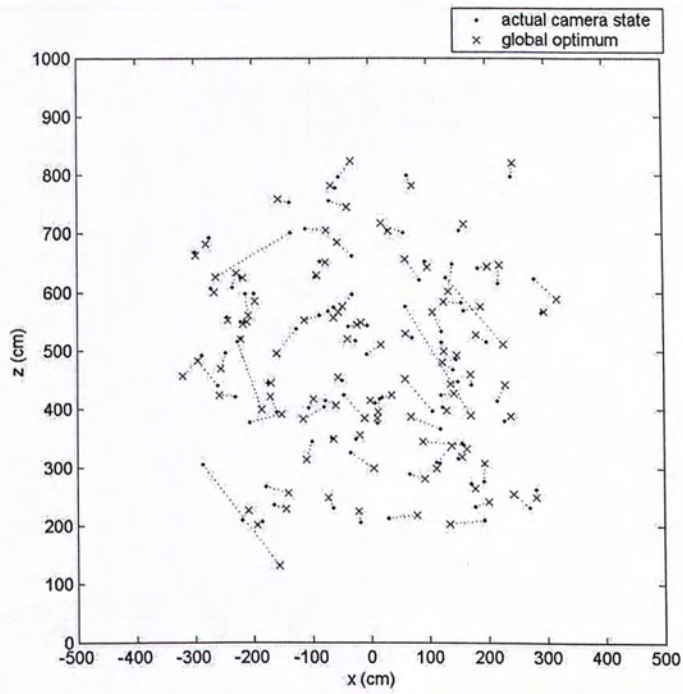


Fig. 6. 19 Differences between actual camera states and global optimums for  $N_m = 5$  and  $\sigma_n = 4$  pixels (Topview).

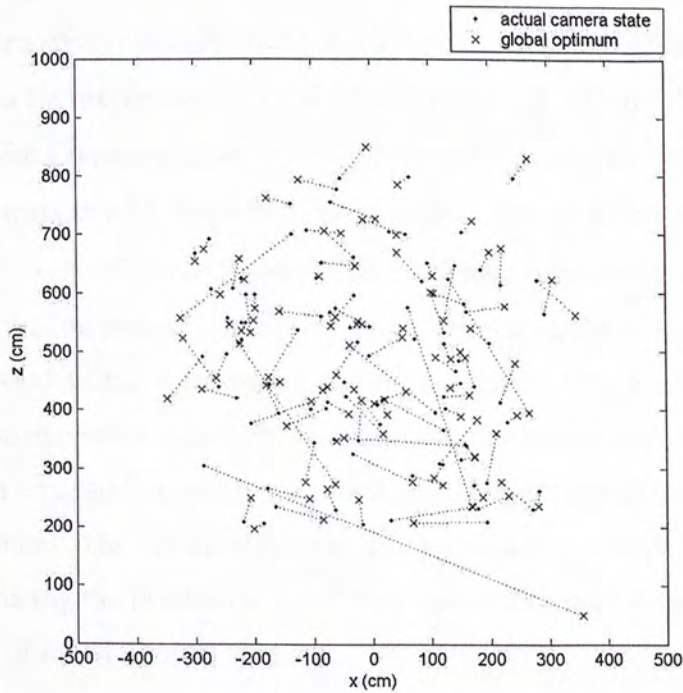


Fig. 6. 20 Differences between actual camera states and global optimums for  $N_m = 5$  and  $\sigma_n = 8$  pixels (Topview).

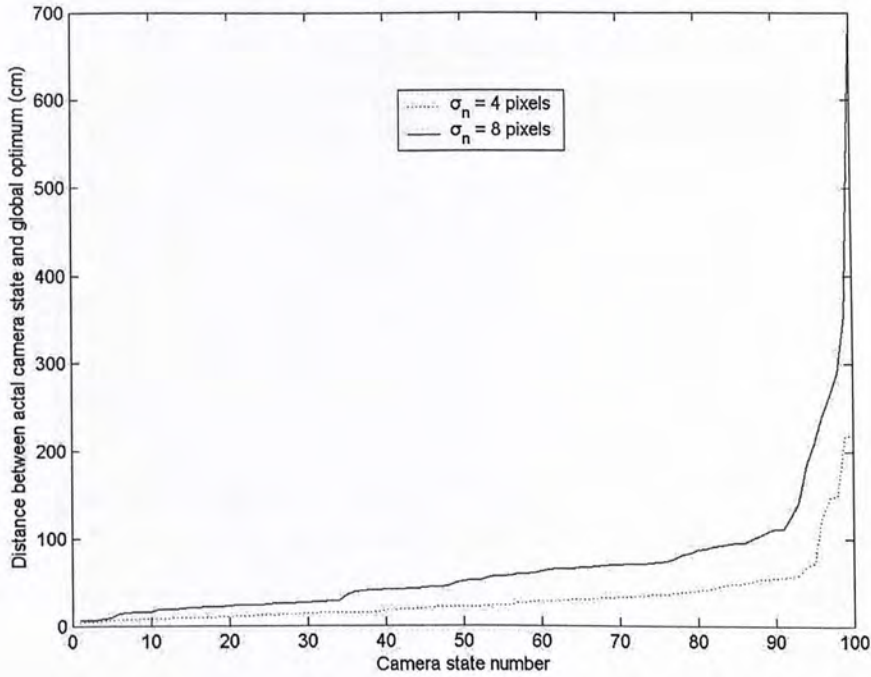


Fig. 6. 21 Sorted distances between the global optimums and the actual camera states for feature points added with Gaussian noise with standard deviation  $\sigma_n = 4$  pixels and 8 pixels.

Comparisons between the results of the average position errors ( $E_p$ ) of the estimated camera states obtained in the experiment 2 (section 6.6) and the  $\Delta r$ , for  $N_m$  varied from 5 to 10, are shown in Fig. 6. 22 and Fig. 6. 23. The results for the feature points added with Gaussian noise:  $\sigma_n = 4$  pixels and 8 pixels are shown in Fig. 6. 22 and Fig. 6. 23 respectively. Seen from these figures, the  $\Delta r$  is larger than the  $E_p$  for different noise levels of feature points. The difference between  $\Delta r$  and  $E_p$  increases along with the feature points' noise level. In experiment 2, the adaptive search space strategy is applied to the localization algorithm. Hence, Fig. 6. 22 and Fig. 6. 23 demonstrate the estimated camera position errors differences between searching for the optimum in an adaptive search space and searching for the global optimum in the entire environment. The results illustrate that the adaptive search space strategy is efficient in reducing the position error of the estimated camera state especially when the noise level of feature points is high.

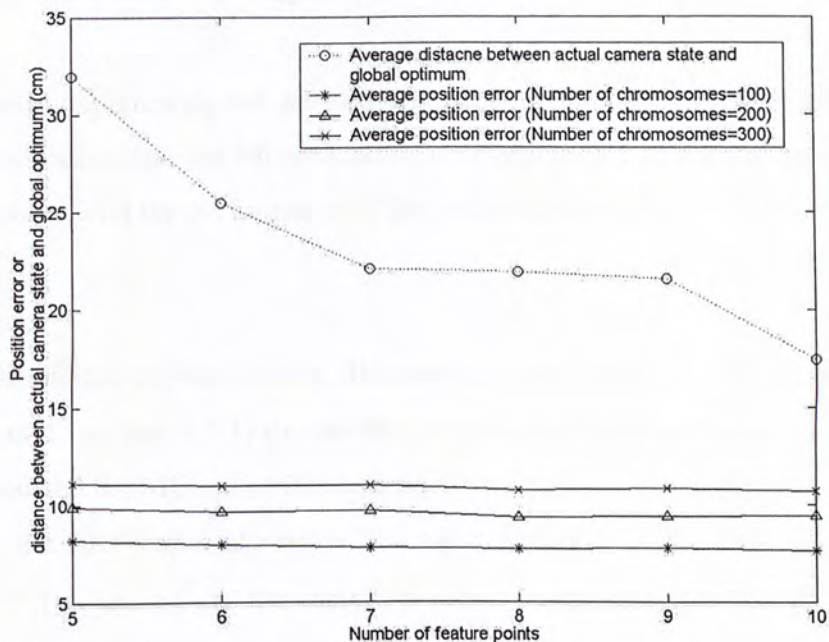


Fig. 6. 22 Average distances between the actual camera states and the global optimums, and average position errors with feature points added with Gaussian noise ( $\sigma_n = 4$  pixels).

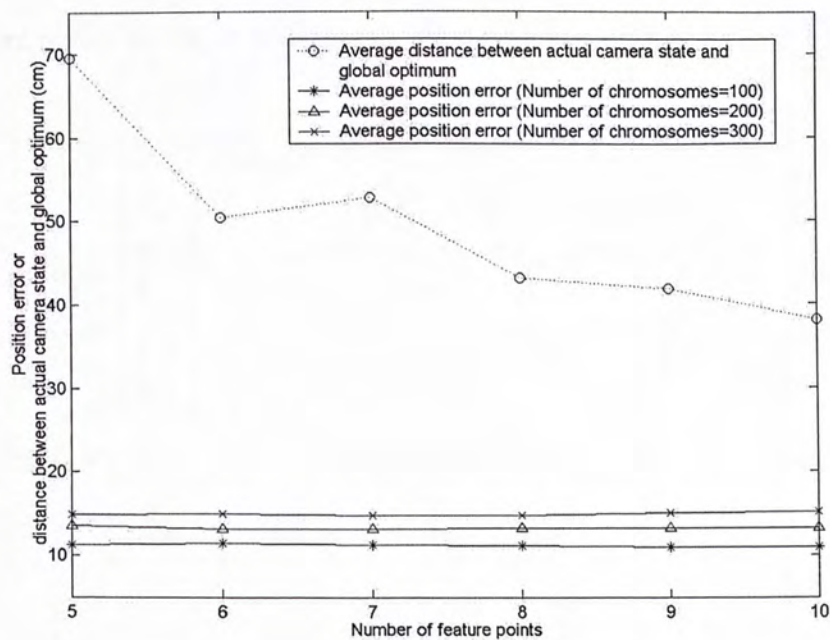


Fig. 6. 23 Average distances between the actual camera states and the global optimums, and average position errors with feature points added with Gaussian noise ( $\sigma_n = 8$  pixels).

## 6.8 Experiment 4 – Comparison with benchmark algorithms

In this experiment, the performance of the Extended Kalman filter (EKF)-based localization and the Monte Carlo Localization (MCL) algorithms are studied and compared with the performance of the proposed algorithm.

### 6.8.1 Setup

In the following simulations, the experimental setups are the same as that of experiment 1 (section 6.5.1) except that the equipped camera is localized using the EKF-based and the MCL algorithms respectively.

For the EKF-based algorithm, the camera state is in the form:  $\xi = [\rho^T, \phi^T]^T$  where  $\rho = [\rho_x, \rho_y, \rho_z]^T$  is the camera position vector and  $\phi = [\phi_x, \phi_y, \phi_z]^T$  is the camera orientation represented by Euler angles. As discussed in section 2.1.1, the state of a discrete-time process is governed by Eq.(2. 1) and the state is related to the measurement by Eq.(2. 2). In the following simulations, the control input  $\mathbf{u}$  in Eq. (2. 1) is composed of translation and rotation magnitudes:  $\mathbf{u} = [trans, rot]$ , and Eq.(2. 1) and Eq.(2. 2) are formulated as Eq. (6. 4) and Eq.(6. 5) respectively. For each frame, five feature points are input to the localization system, i.e.  $N_m = 5$ .

$$\xi_k = \begin{bmatrix} \rho_{x,k} \\ \rho_{y,k} \\ \rho_{z,k} \\ \phi_{x,k} \\ \phi_{y,k} \\ \phi_{z,k} \end{bmatrix} = \begin{bmatrix} \rho_{x,k-1} - trans_{k-1} \times \sin(\phi_{y,k-1}) + w_1 \\ p_{y0} + w_2 \\ \rho_{z,k-1} + trans_{k-1} \times \cos(\phi_{y,k-1}) + w_3 \\ w_4 \\ \phi_{y,k-1} + rot_{k-1} + w_5 \\ w_6 \end{bmatrix} \quad (6.4)$$

where  $\mathbf{w} = [w_1, w_2, w_3, w_4, w_5, w_6]^T$  is the process noise vector.

$$\mathbf{z}_k = \begin{bmatrix} m_{u,1} \\ \vdots \\ m_{u,N_m} \\ m_{v,1} \\ \vdots \\ m_{v,N_m} \end{bmatrix} + \begin{bmatrix} v_1 \\ \vdots \\ \vdots \\ \vdots \\ v_{2N_m} \end{bmatrix} \quad (6.5)$$

$$\begin{bmatrix} m_{u,i} \\ m_{v,i} \end{bmatrix} = \begin{bmatrix} c_x + \frac{F}{p_u} \times \frac{h_{x,i}}{h_{z,i}} \\ c_y + \frac{F}{p_v} \times \frac{h_{y,i}}{h_{z,i}} \end{bmatrix}$$

$$\begin{bmatrix} h_{x,i} \\ h_{y,i} \\ h_{z,i} \end{bmatrix} = \mathbf{R} \times (\mathbf{f}_i - \mathbf{p}_{k-1})$$

where  $\mathbf{v} = [v_1, \dots, v_{2N_m}]^T$  is the measurement noise and  $\mathbf{R}$  is obtained by substituting  $[\theta_x, \theta_y, \theta_z]$  in Eq.(3. 10) with  $\phi_{k-1}$ .

In order to study the performance of the MCL algorithm, a MCL-based algorithm proposed by [Rofer and Jungel, 2003] is employed in the simulation of camera state localization. According to the MCL algorithm mentioned in section 2.2.1, the camera state represented by a particle is in the form:  $\mathbf{l} = [x, y, \theta]$  where  $[x, y]$  is the camera position vector and  $\theta$  is the camera orientation. The number of particles  $N_c$  is set to 200. Similar to the setting of the EKF-based algorithm simulation, five feature points per frame is input to the localization system, i.e.  $N_m = 5$ . In the sensor reading phase, each particle is weighted by the probability  $P$ . Suppose the measured and expected bearings of the  $i^{th}$  input feature point for a certain particle position are  $\omega_m^i$  and  $\omega_e^i$  respectively,  $P$  is computed by:

$$P = \prod_{i=1}^5 \varepsilon_i$$

$$\text{where } \varepsilon_i = \begin{cases} e^{-50d^2} & , \text{if } d < 1 \\ e^{-50(2-d)^2} & , \text{otherwise} \end{cases} \quad (6.6)$$

$$\text{and } d = \frac{|\omega_m^i - \omega_e^i|}{\pi}$$



---

In order to calculate the camera state at a certain time step, the search space is divided into  $10 \times 10 \times 10$  cells and each particle is assigned to one of these cells. The  $2 \times 2 \times 2$  sub-cube that contains the maximum number of particles is considered. The mean of all the particles that belongs to this sub-cube is the estimated camera state.

### 6.8.2 Results

The results of localizations using the EKF-based and MCL algorithms are shown in Fig. 6. 24 and Fig. 6. 25 respectively. The estimated camera paths show in Fig. 6. 24 and Fig. 6. 25 diverge from the actual camera path soon after the camera is departed from the starting point, i.e.  $[0, 0]$ . Moreover, both estimated camera paths do not tend to return to the actual path after the divergences.

The position errors (2D) of the camera paths estimated by the EKF-based and MCL algorithms are shown in Fig. 6. 26 and Fig. 6. 27 respectively. In these figures, the position error increases along with the increment of frame number (time). This indicates that the robot track will probably lost when the EKF-based or MCL algorithm is employed to the localization of the legged robot which is navigated in a relatively large environment. Moreover, the performance of the EKF-based and MCL algorithm is expected to be degraded if the input images are noisy.

The position errors (2D) of the EKF-based and MCL algorithm estimated paths averaged over the number of frames are 47 cm and 55 cm respectively. On the other hand, the average position errors (3D) of the proposed algorithm obtained in experiment 1 (Fig. 6. 4(a)) are varied from 3.4 cm to 4.3 cm.

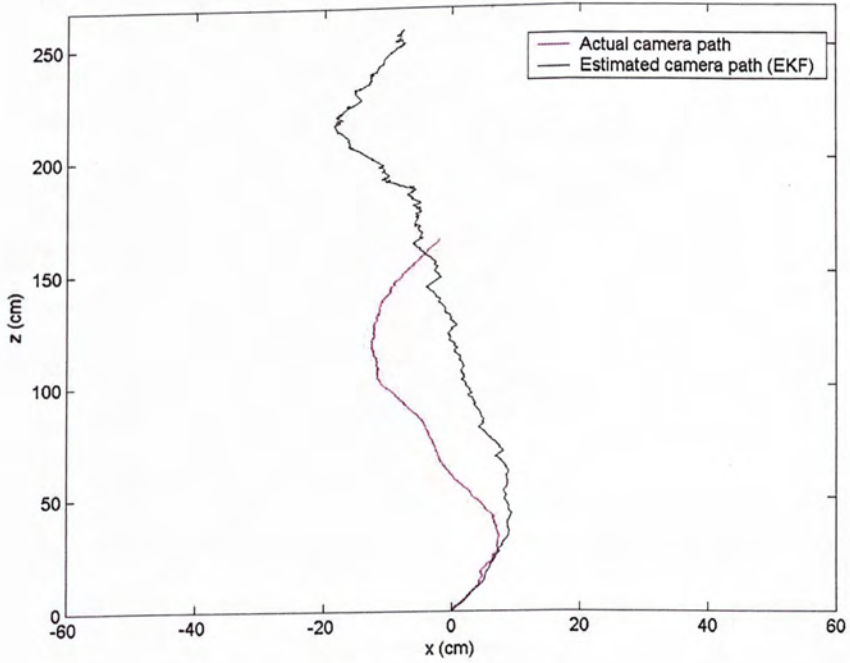


Fig. 6. 24 Localization results of the EKF-based localization algorithm (Topview).

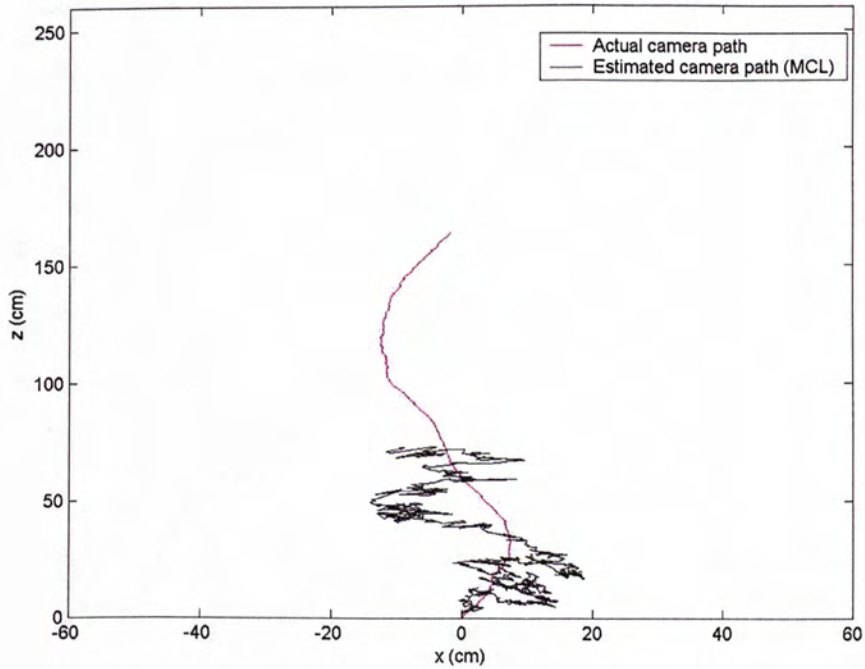


Fig. 6. 25 Localization results of the MCL algorithm (Topview).

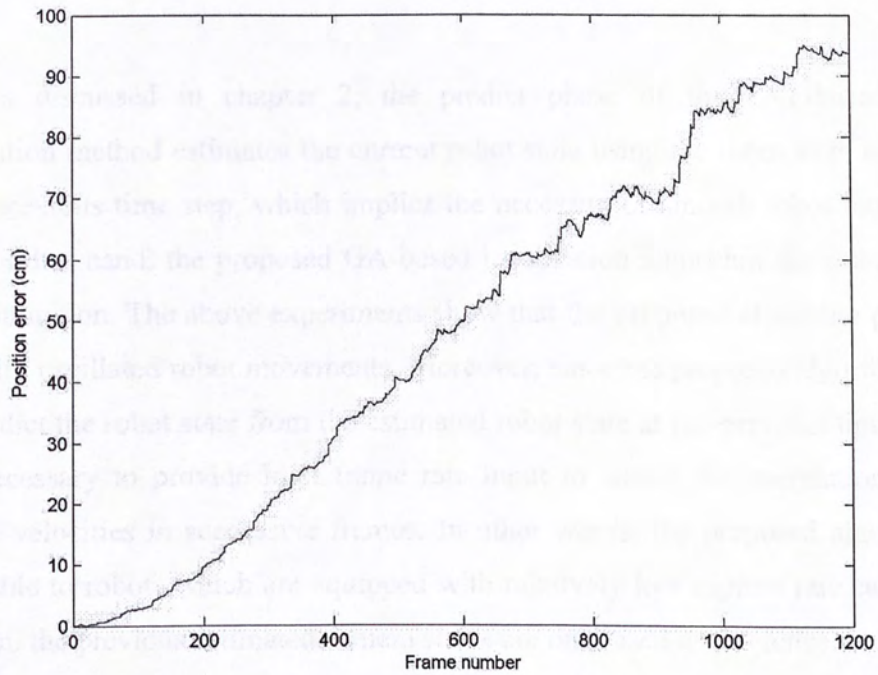


Fig. 6. 26 Position errors of localization results using the EKF-based algorithm.

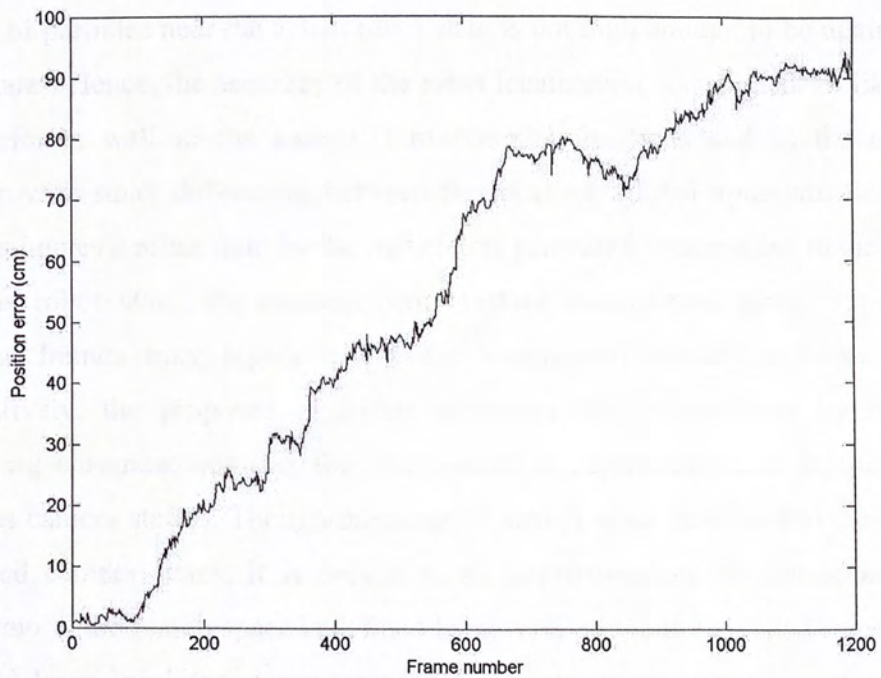


Fig. 6. 27 Position errors of localization results using the MCL algorithm.

## 6.9 Discussions

As discussed in chapter 2, the predict phase of the EKF-based visual localization method estimates the current robot state using the robot state estimated at the previous time step, which implies the necessity of smooth robot movement. On the other hand, the proposed GA-based localization algorithm does not require this assumption. The above experiments show that the proposed algorithm performs well with oscillated robot movements. Moreover, since the proposed algorithm does not predict the robot state from the estimated robot state at the previous time step, it is unnecessary to provide high frame rate input to ensure the correlation among camera velocities in successive frames. In other words, the proposed algorithm is applicable to robots which are equipped with relatively low capture rate camera. In addition, the previous estimated camera states are only used as reference for defining the search space. It is a rough approximation which aims to increase the efficiency of the searching process and to give a boundary of reasonable camera states.

MCL is a probabilistic method that the probability density function (p.d.f.) of a robot state is represented by particles. When landmarks are far from the robot, the density of particles near the actual robot state is not high enough to be distinct from other states. Hence, the accuracy of the robot localization is degraded. Unlike MCL, GA performs well in the cases: 1) multimodal functions and 2) the objective function with small differences between the local and global optimum. Besides, as MCL estimates a robot state by the particle set generated with respect to the p.d.f. of previous robot state, the measurement (feature coordinates) errors appeared in previous frames may highly affect the localization results in future frames. Alternatively, the proposed algorithm estimates the camera state by randomly generating chromosomes (i.e. the distribution of chromosomes is independent to previous camera states). Though the range of search space is related to the previous estimated camera states, it is only a rough approximation, as mentioned above. Furthermore, the search space is defined by several previous estimated camera states to avoid large localization error caused by measurement errors in a frame from affecting the localization results in future. Therefore, the estimated camera state by the proposed algorithm is insensitive to the previous measurement errors unless

---

large localization errors are involved in all previous reference camera states. In addition, the adaptive search space strategy weakens the influence of measurement errors in the current frame on localization results by bounding the search within a reasonable range. It is especially sufficient when the feature points are involved with high level of noise.

Since similar research on 6D localization for legged robots was not found, localization results of some 3D localization methods for legged robots are reported. The accuracy of a MCL-based localization method for a legged robot in the RoboCup environment proposed by [Lenser and Veloso, 2000] is reported to be: average 2D position error approximately 9 cm and average orientation error approximately  $14^\circ$ . The accuracy of another MCL-based method proposed by [Sridharan *et al.*, 2005] is reported to be: average 2D position error approximately 9 cm and average orientation error approximately  $3^\circ$ . It is inappropriate to make any comparison between these algorithms and our proposed algorithm as these experiments are 3D localizations and their experimental setups and camera qualities are not identical to our experiments.

In addition, we should point out that the robot control commands are unknown to the proposed algorithm. In each camera state estimation, the search space is defined as there is no information about the moving direction of the robot. The possible camera state becomes the maximum moving distance in all directions (see section 5.6). However, we expect that the performance of the proposed algorithm can be improved if the control commands are also input to the localization algorithm. With the information from control commands, the search space size can be further reduced, and the accuracy and efficiency of the proposed algorithm can be improved.

Another possible way to improve the proposed algorithm is an appropriate feature point selection strategy. In the above experiments, the feature points used for localization are randomly selected from all feature points appeared in an image. It is expected that the feature point of near landmark is more reliable than that of far landmark especially for noisy image. Due to lens distortion, feature points near the center of the captured image should be more reliable than those at the edges of the image. Hence, a proper feature selection strategy might help to improve the localization accuracy.

## 6.10 Summary

In this chapter, the performance of the proposed algorithm is examined by several experiments. In the first and second experiments, simulations of continuous camera localization using feature points with low and high level of noise were performed. The results of these experiments show that the accuracy and efficiency of the proposed algorithm is satisfactory even though the feature points are noisy. In the third experiment, the effect of the noisy feature points on the difference between global optimum and actual camera state is demonstrated by randomly generating 100 camera states and adding different levels of noise to their feature points. The results show that the difference between the global optimum and the actual camera state is significant, especially when the noise level is high. This indicates that the adaptive search space strategy not only improves the efficiency of the localization, but also weakens the influence of the noisy feature point on the estimated camera state accuracy by limiting the search in a reasonable range.

---

## ***Chapter 7 – Conclusion***

This chapter concludes the thesis by summarizing the main development and results.

The thesis addresses the needs of an accurate and efficient localization method for small home-use robot pet (legged robot) which is equipped with a single low-resolution camera as the only sensor. The main challenges of vision-based localization for legged robot include: 1) the rapid changes of robot velocity which cause the sensor and its collected data fluctuated, 2) compared with wheeled robot, the high degree of freedom of legged robot increases the complexity of the localization problem and 3) camera has limited field of view and images from single camera are lack of depth information. Thus, an ideal vision-based localization algorithm for legged robot should be adapted to the oscillated sensor data and able to estimate the robot state accurately and efficiently. Moreover, it should be able to cope with both robot tracking and global localization, and able to autonomously shift between them.

The state of the art of the vision-based localization includes the Extended Kalman Filter (EKF)-based localization and the Monte Carlo Localization (MCL), are reviewed in chapter 2. Their advantages and disadvantage are also discussed.

The vision-based localization problem of legged robot or high-dimensional movement robot is defined as: given the feature points of an image captured by the camera equipped on a robot at a certain instant and the landmark positions correspond to the feature points, trying to estimate the robot state (position and orientation) at that instant. We assume that the robot navigation environment consists of a set of landmarks with known positions and some of these landmarks are viewed by the camera at each time step. By estimating the robot state repeatedly at each time step, the robot path can be tracked. Based on the defined problem, we formulate the vision-based localization for high-dimensional movement robot as an optimization in chapter 3.

The camera state at the current instant can be obtained by optimizing the objective function formulated in chapter 3. The mechanism of some common search algorithms including calculus-based, enumerative and stochastic algorithm are

presented in chapter 4. The limitations and the efficiencies of these algorithms are discussed. As the proposed objective function is a multimodal and non-linear function, the stochastic algorithm is the best solution to the optimization of the objective function. A well-known algorithm in the stochastic approach class, genetic algorithm (GA), is employed in the proposed localization algorithm.

The mechanism of the genetic algorithm is introduced in chapter 5. The details of the employment of GA including chromosome formation, fitness function, genetic operator, selection scheme and search space which targets the vision-based localization problem are presented. Based on the basic mechanism of GA, we proposed several approaches to improve the efficiency of the proposed localization algorithm. First, the number of optimum in the search space is reduced by redefining the camera orientation vector. The redundant Euler angles representation is replaced by the unique unit quaternion representation. Afterward, we modified the objective function that is formulated in chapter 3, and three mutually independent orientation genes are formed by modeling three of the parameters in the quaternion as a point in a unit sphere. The strategy reduces the number of optimum while maintains the search space dimensions. Second, a search space defining method, called adaptive search space strategy, is proposed with the assumption that the maximum distance between robot positions at successive time steps are limited by the robot's walking speed. The strategy increases the searching efficiency by reducing the search space size to part of the navigation area while prevents the actual robot state from excluding from the search space. Moreover, the strategy is insensitive to localization error at the past as the localization results at previous few time steps are used as reference. Besides increasing the searching efficiency, the adaptive search space strategy is able to improve the accuracy of the estimated camera state especially when the input feature points involve high level of noise. As the global optimum is significantly different from the actual robot state when the feature points is noisy, the adaptive search space strategy weakens the influence of noisy feature points on the accuracy of localization by bounding the search within a reasonable range. Lastly, based on the assumption that the chromosomes are fallen into the global lobe after half of the maximum number of generation  $T_g$ , the computational cost is saved by reducing the population size to half after the  $(T_g/2)^{th}$  generation.



---

The performance of the proposed algorithm is tested by experiments and the results are shown in chapter 6. The results of the simulation of continuous camera localization in an area of  $1000\text{cm} \times 1000\text{cm}$  for 60 seconds show that the proposed algorithm is able to track the camera state accurately and efficiently. In the second experiment, simulation similar to the first one is performed except that the input feature points are added with noise. The results show that the proposed algorithm is insensitive to noise. Moreover, the ability of the adaptive search space strategy in weakening the influence of noisy feature points on the estimated camera state accuracy is further confirmed by the third experiment. Experimental results showed that difference between the state of global optimum and the actual robot state is large when the noise level of feature points is high. Hence, the results prove that without applying the adaptive search space strategy, large error can be induced in the estimated camera states when the feature points are noisy.

To summarize, our main contributions include: 1) demonstration of genetic algorithm as a possible solution to vision-based localization for small home-use robot navigated in a practical environment, 2) the proposed algorithm does not require the assumption of smooth robot movement so that it is applicable to walking robots or robots with rapid velocity change, 3) compared with the EKF-based localization algorithms, the proposed algorithm can tolerate low frame rate input, 4) the proposed algorithm is insensitive to localization errors in previous estimations compared with MCL, which avoids error propagation from previous estimations, 5) the adaptive search space strategy weakens the influence of measurement errors in current frame on the localization result by bounding the search within a reasonable range and 6) development of a fundamental localization system based on the proposed algorithm, which can be used for further investigations, e.g. global localization and feature point reliability analysis, on the proposed algorithm.

## References

- [Bailey *et al.*, 1999] Bailey, T., Nebot, E. M., Rosenblatt J. K. and Durrant-Whyte, H. F., "Robust distinctive place recognition for topological maps," in *Proc. of Int. Conf. on Field and Service Robotics*, pp. 347-352, 1999.
- [Bonci *et al.*, 2005] Bonci, A., Ippoliti, G., La Manna, A., Longhi, S. and Sartini, L., "Sonar and video data fusion for robot localization and environment feature estimation," in *Proc. of the 44th IEEE Conf. on Decision and Control, and the European Control Conf.*, pp. 8337-8347, 2005.
- [Chou *et al.*, 2004] Chou, H., Traonmilin, M., Ollivier, E. and Parent, M., "A simultaneous localization and mapping algorithm based on Kalman filtering," in *IEEE Intelligent Vehicles Symposium*, pp. 631-635, 2004.
- [Davison, 2003] Davison, A. J., "Real-time simultaneous localisation and mapping with a single camera," in *Proc. of the Ninth IEEE Int. Conf. on Computer Vision*, pp. 1403-1410, 2003.
- [Davison *et al.*, 2003] Davison, A. J., Mayol, W. W. and Murray D. W., "Real-time localization and mapping with wearable active vision," in *Proc. of the second IEEE and ACM Int. Symposium on Mixed and Augmented Reality*, pp. 18-27, 2003.
- [Dellaert *et al.*, 1999] Dellaert, F., Fox, D., Burgard, W. and Thrun, S., "Monte Carlo Localization for mobile robots," in *Proc. of IEEE Conf. on Robotics and Automation*, vol. 2, pp. 1322-1328, 1999.
- [Dissanayake *et al.*, 2000] Dissanayake, G., Durrant-Whyte, H. and Bailey T., "A computationally efficient solution to the simultaneous localisation and map building (SLAM) Problem," in *Proc. of IEEE Int. Conf. on Robotics and Automation*, vol. 2, pp. 1009-1014, 2000.
- [Dissanayake *et al.*, 2001] Dissanayake, M. W. M. G., Newman, P., Clark, S., Durrant-Whyte, H. F. and Csorba, M., "Solution to the Simultaneous Localization and Map Building (SLAM) Problem," in *IEEE Trans. on Robotics and Automation*, vol. 17, issue 3, pp. 229-241, June 2001.
- [Fox *et al.*, 1999] Fox, D., Burgard, W., Dellaert, F. and Thrun, S., "Monte Carlo Localization: Efficient position estimation for mobile robots," in *Proc. of the Sixteenth Nat. Conf. on Artificial Intelligence*, pp. 343-349, 1999.

- 
- [Gassmann *et al.*, 2005] Gassmann, B., Zacharias, F., Zollner, J. M. and Dillmann, R., "Localization of walking robots," in *Proc. of IEEE Int. Conf. on Robotics and Automation*, pp. 1471-1476, 2005.
- [Graf *et al.*, 2004] Graf, B., Hans, M. and Schraft, R. D., "Mobile robot assistants," in *IEEE Robotics & Automation Magazine*, vol. 11, issue 2, pp. 67-77, June 2004.
- [Gross *et al.*, 2001] Gross, H.-M., Boehme, H.-J. and Wilhelm, T., "Contribution to vision-based localization, tracking and navigation methods for an interactive mobile service-robot," in *IEEE Conf. on Systems, Man, and Cybernetics*, vol. 2, pp. 672-677, 2001.
- [Guivant and Nebot, 2001] Guivant, J. E. and Nebot, E. M., "Optimization of the simultaneous localization and map-building algorithm for real-time implementation," in *IEEE Trans. on Robotics and Automation*, vol. 17, issue 3, pp. 242-257, June 2001.
- [Jeong and Lee, 2005] Jeong, W.Y. and Lee, K.M., "CV-SLAM: a new ceiling vision-based SLAM technique," in *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 3195-3200, 2005.
- [Kalman, 1960] Kalman, R. E., "A New Approach to Linear Filtering and Prediction Problems," in *Trans. of the ASME – Journal of Basic Engineering*, vol. 82, series D, pp. 35-45, 1960.
- [Karlsson *et al.*, 2005] Karlsson, N., Di Bernardo, E., Ostrowski, J., Goncalves, L., Pirjanian, P. and Munich, M. E., "The vSLAM algorithm for robust localization and mapping," in *Proc. of IEEE Int. Conf. on Robotics and Automation*, pp. 24-29, 2005.
- [Lenser and Veloso, 2000] Lenser, S. and Veloso, M., "Sensor resetting localization for poorly modelled mobile robots," in *Proc. of IEEE Int. Conf. on Robotics and Automation*, vol. 2, pp. 1225-1232, 2000.
- [Liu *et al.*, 2006] Liu, P. R., Meng, M., Liu, P., X., Tong, F. L. and Chen, X. J., "A telemedicine system for remote health and activity monitoring for the elders," in *Telemedicine and e-Health*, vol. 12, issue 6, pp. 622-631, 2006.
- [Lowe, 2004] Lowe, D. G., "Distinctive image features from scale-invariant keypoints," in *Int. J. of Computer Vision*, vol. 60, issue 2, pp. 91-110, 2004.

- 
- [Rofer and Jungel, 2003] Rofer, T. and Jungel, M., "Vision-based fast and reactive Monte-Carlo Localization," in *Proc. of IEEE Int. Conf. on Robotics and Automation*, vol. 1, pp. 856-861, 2003.
- [RoboCup] RoboCup Official Site: <http://www.robocup.org>.
- [Sridharan *et al.*, 2005] Sridharan, M., Kuhlmann, G. and Stone, P., "Practical vision-based Monte Carlo Localization on a legged robot," in *Proc. of IEEE Int. Conf. on Robotics and Automation*, pp. 3366-3371, 2005.
- [Wang *et al.* 2005] Wang, H., Yuan, K., Zou, W. and Zhou, Q., "Visual odometry based on locally planar ground assumption," in *Proc. of IEEE Conf. on Information Acquisition*, pp. 59-64, 2005.
- [Welch and Bishop, 2004] Welch, G. and Bishop, G., "An introduction to the Kalman filter," in *The University of North Carolina at Chapel Hill, Technical Report 95-041*, April 5, 2004.
- [Ueda *et al.*, 2003] Ueda, R., Arai, T., Asanuma, K., Kamiya, S., Kikuchi, T. and Umeda, K., "Mobile robot navigation based on expected state value under uncertainty of self-localization," in *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, vol. 1, pp. 473-478, October 2003.



CUHK Libraries



004461282