Visually Guided Obstacle Detection and Avoidance for Legged Robot

CHOW Ying-ho

Department of Mechanical and Automation Engineering

The Chinese University of Hong Kong

A thesis submitted in partial fulfillment of the requirement for the degree of

Master of Philosophy in Engineering

July 2000

The work described in this thesis was partially supported by a grant from the Research Grants Council of Hong Kong Special Administrative Region, China

RGC Ref. No. CUHK4169/98E

摘　要

此論文描述一個障礙物探測及避免踫撞系統，此系統是利用機械人視覺技術及模糊邏輯控制的。此系統的設計是特別地爲了應用在一個富挑戰性的流動機械人平台——爬行式機械人。爬行式的機械人在移動時會比使用車輪的機械人，在拮取到的圖象資料上造成更大量的振動，一般應用於車輪機械人探測障礙物的方法皆不適用於此情況。此論文提出新的方法解決這問題。

此論文應用了，立體空間裏平面所引致的，影像與影像之間的線性映射關係來探測地面上的障礙物，我們稱呼這種映射關係爲單對應性。再透過實時計算這不斷改變的映射，從不符合這映射的地方找出障礙物。跟着把障礙物圖像再映射到表示地面的近距離局部性地圖(簡稱 LOM)上。但是在圖像的計算上會造成時間的延遲，所以在計算應作出的動作前先要預測實時的 LOM。最後，根據 LOM 的資料，配合模糊邏輯控制器作出適當的動作指令去控制機械人。

在實際的測試中發現，這系統能有效地在實時的、充滿圖案的環境裏，避開地面上的牆壁、在轉角處轉彎及在末處回頭。

Abstract

An obstacle detection and avoidance system that uses stereo-vision is described. The system is designed for a challenging mobile platform: a legged robot, whose locomotion is not as smooth as the wheeled robots' and thus induces more disturbances to the visual data. The concept of the image-to-image mapping named homography induced by the ground is exploited for detecting obstacles. Such a mapping is allowed to change over time as the legged motion takes place. Obstacles can thus be detected in terms of a 2D distribution on the ground we call the Local Obstacle Map (LOM). Based on the LOM, which is further time-delay compensated, a fuzzy control mechanism is then used to command the robot motion.

Experiments in textured environments show that the system is effective for autonomous and real-time navigation along straight paths, curved corridors, avoiding walls on two sides, making turns around corners, and withdrawing from dead-ends. The results also show that the autonomous navigation is possible even with large delays between image capture and command determination.

Acknowledgments

I would like to thank my supervisor, Prof. Ronald Chung, for providing advice, insight, and guidance along the way. He has been a great supervisor, giving generous support and encouragement. For my colleagues, it's been a remarkable experience working with you all. I would like to thank all the people in the Department of Mechanical and Automation Engineering who have provided me with their support.

Finally, I'd like to extend my thanks and gratitude to my family and friends for always being there when I needed them.

Table of Contents

List of Figures

Chapter 1

Introduction

Though legged robots have distinct advantages over wheeled robots like the greater capability of climbing over stairs and a higher energy-efficiency, research on them has been focused mostly on their balancing and locomotion. In fact, their obstacle detection, obstacle avoidance, and navigation planning have characteristics of their own quite unlike those of wheeled counter parts, and deserve separate studies, such as their locomotion is not as smooth and induces more disturbances to the visual data acquirable by cameras mounted on them and thus a challenging task for obstacle detection and avoidance systems based on visual information. This work serves to contribute to the study of how a legged robot mounted with a stereo pair of cameras can detect and avoid obstacles using the image data.

1.1 Objectives – Visual Navigation for Legged Robots

Visual information has a special role in the long-term planning of a navigation task, but undeniably, it is used most of the time for watching the immediate or near-immediate steps. A substantial part of the image data is thus about places within a few steps of the cameras or the robot they are mounted on. For such image data, it is just logical from the lever principle and it is confirmed by empirical observations that a single gait of the robot would cause tremendous disturbance to the data. For this reason, we believe the optical flow based approaches widely proposed for wheeled robots [9][15] are not well suited for legged robots. In this work, we would exploit not the temporal attribute of the image data, but its spatial attribute, to probe the 3D nature of the surroundings. More precisely, we would use not the image motion data but the stereo data to detect and avoid obstacles, as the spatial configuration of the stereo cameras can practically be made rigid and invariant with respect to the robot at all time.

In this thesis, we make the assumptions that at any particular instant the robot is moving on a surface, we call ground which is approximately flat at least in the immediate vicinity of the robot, and in the vicinity of the robot anything above and below the locally ground plane, is an obstacle to avoid. We also assume that the two cameras are placed symmetrically about the robot's forward-moving direction, looking out to the front left and the front right of the robot, though an extreme accuracy is not necessary and in fact the cameras were mounted only manually in our implementation. There has been a number of work that uses the "ground" assumption (e.g.,[20][29]), but they use it in the context of obstacle detection from a particular stereo image pair. Here we use it for obstacle detection of a moving robot, and in particular of a moving legged robot. More specifically, we allow the mapping between the stereo images induced by the ground be changing over time due to the changing geometry between the robot and the ground.

Like other obstacle avoidance algorithms, most of the previous work that use the concept of obstacle map, either the obstacle map is assumed given or acquired from a range sensor (most often the ultrasound-sensing type). Range sensors, in comparison with camera, have several disadvantages [29]. It is obvious that the resolution of the data is limited to the number of sensors mounted on a robot. The other disadvantage is caused by its reflection-based nature, surfaces of degeneration may miss orientations (or orientations close to those degeneracies) with respect to the sensors, and this happens quite often when the robot is at the corner of a corridor. Moreover, inter-reflections among surfaces may dilute the accuracy of the range sensors. So, ultrasonic sensors are only limited for close-ranges in order to avoid misleading inter-reflections. Since our work is aimed at acquiring the local obstacle map not from range sensors but from cameras which are passive sensors that do not send out signals and interfere with the activities in the surroundings. Using the same sensor, long-range and closer-range obstacles can also be detected and the obstacle map can be built in higher resolutions.

1.2 Summary of Results

Experiments in textured environments show that the system is effective for autonomous and real-time navigation along straight paths and curved corridors while avoiding walls on the two sides, making turns at corners, and withdrawing from dead-ends with image resolution of 384 by 288 pixels. The results also show that the navigation is performed without the affect of the large delay between images capture and command determination.

1.3 Hardware Issues

A complete navigation system has been implemented and tested with real environments. It consists of a six-legged robot of the size about 13"x12"x10", a stereo pair of black-and-white miniature cameras with wide angle lenses mounted on it, and a Pentium II remote-brain connected to via wireless video transmission. The remote brain is where most of the calculations take place.

In Chapter 3, Fig.3 shows the robot. The robot is not a sophisticated one; it has each leg activated by two RC servos and is embedded with a Motorola 68332 microcontroller that drives all the servos.

1.4 Contributions

The contributions of this work include:

- An obstacle detection and avoidance algorithm that needs no explicit 3D reconstruction.
- A fuzzy control mechanism that commands the robot motion based on a 2D obstacle distribution only.
- A real-time implementation of a complete navigation system.

1.5 Organization of the Thesis

The thesis is organized as follows: Chapter 2 reviews some related work contributed in obstacle detection and control strategies. Chapter 3 briefly describes how the system work. Chapter 4 describes the detection of obstacles in the stereo images and the construction of a local obstacle map. Chapter 5 describes the gait control and the fuzzy controller used for obstacle avoidance. Chapter 6 is an analysis of the computation complexity of the system. Chapter 7 shows the mobile robot system and the hardware implementation. Chapter 8 reports how an implementation of the system performed in a textured, locally planar environment full of walls, small hills, corners, and dead-ends. Chapter 9 presents a conclusion and future work.

Chapter 2

Previous Work

This thesis draws on earlier work in the fields of computer vision and robot navigation control. Presented below is a brief discussion of some of the work from these fields.

2.1 Vision Based Navigation

Obstacle detection and avoidance form a basic competence for mobile robots to accomplish tasks autonomously in changing environments. Obstacle detection involves acquisition and processing of the sensor data, while obstacle avoidance is primarily mobile robot path planning and control. Mobile robots most often encounter ground plane obstacles, so ground plane obstacle detection has become a key issue for mobile robot applications and has been addressed by many researchers [18], [38] and [48].

A number of active ranging sensors have been used for obstacle detection, including sonar [23][28] and laser ranging [14]; but all these sensors are suffering different kinds of disadvantages, such as, short sensing range, reflection and low angular resolution. To overcome such problems, several investigations have used vision: monocular vision with motion analysis [7][27][35], stereo vision [18][53], trinocular vision [1] and even multiple camera systems [24]. Many researchers have put much effort to develop a general theoretical framework from which a broad range of problems could be addressed, such as obstacle detection via 3D reconstruction. However, such generality that not often the best and directly solve the problem. Ground plane obstacle detection is one of the types.

2.1.1 Homography

Recently an image-to-image mapping named homography was introduced for computer vision by Faugeras [16][17]. A homography is a mapping induced by a planar surface in 3D. While corresponding features in the stereo images projected by the same plane ought to satisfy the mapping the plane induces, features not on that plane generally do not. The concept of homography has been used primarily for three-view problems, namely how two fully corresponded views of a scene can be used for generating a new view, and how two fully corresponded views of an object can be used to localize the object in any third view, in both cases without the need of an explicit 3D reconstruction of the involved scene or object.

Homography has been used in several pieces of work. However, most of the uses have been limited to only three-view problems and the recognition of an object in a third view using two fixed views as reference. In this thesis, the concept of Homography is shown that it can be applied to the field of obstacle detection in robot navigation.

In Fig.1, an example of the simple obstacle detection, which uses homography is demonstrated. We can see that in the final difference image, other than the unwarpable areas, the textures on the ground are subtracted nearly to zero. Only some of the object above the ground is visible. This example easily shows that it works in finding non-ground areas in an image.

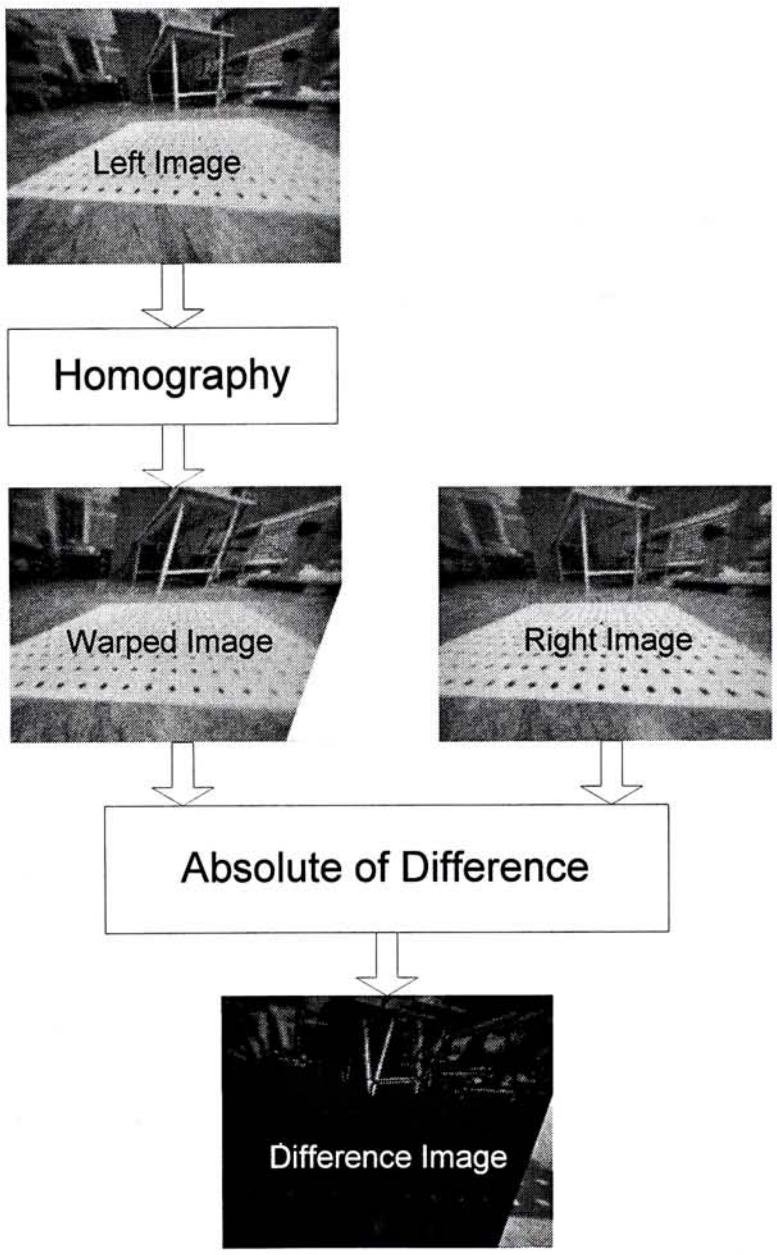


Fig.1 Principle of homography based ground plane obstacle detection

2.1.2 Ground Plane Obstacle Detection

Ground plane obstacle detection has been investigated in a variety of ways from reconstructing 3D scene structure to using direct observables from cameras such as optical flow, disparities, and the rate of change of disparities without visual reconstruction. By reconstructing the ground plane in 3D, we are able to detect ground plane obstacles using a parallel computing that explicit 3D reconstruction is not necessary for vehicle navigation, so that not only can precise camera calibration be avoided but also a more reliable control input variables can be obtained in the obstacle avoidance control system.

Most approaches to visual obstacle detection exploit motion cues for locating obstacles. Furthermore, an assumption that is often made is that vehicle motion is confined to a surface that is either planar or can be approximated locally by planes [40], [19], [20], [6], [29], [47], [52] and [31].

For examples, Santos-Victor and Sandini [40] employed the estimated normal flow field with an uncalibrated camera and detect obstacles lying on a planar floor by performing an inverse perspective transformation that maps the normal flow onto a horizontal (parallel to the floor) plane. However, their methods use an approximate parametric model of the flow generated by the ground plane, deals with outliers in an ad hoc manner and requires the camera to remain in a fixed position relative to the vehicle. Fornland [19] uses the normal flow field measured from a camera moving parallel to the ground plane to derive a linear equation relating motion parameters to the spatio-temporal derivatives of the image intensity function. Obstacles are then detected as the outliers of a robust fit estimated by RANSAC over the image points.

Gaspar et al in [20] is the first approach to use the concept of homography to detect obstacle by stereovision. In their paper, they verified the linear projection induced by a 3D plane between stereo image pair. In addition, one of the features in their work is the configuration of the experiment. They used mirrors to divide the view into two halves in order to have the same brightness and contrast in both view.

Bertozzi et al [6] proposed another stereo-based obstacle detection approach. They assumed that the fixed geometry of the ground with respect to the vehicle induces two fixed homographies from stereo images to top-viewed images. Stereo images are then warped to two top-viewed images. After comparison, the obstacles are detected in the top view.

Li and Brady in [29] have extended the homography based obstacle detection method to an active stereo platform. They called the homography mapping as ground plane transformation. In their approach, they decomposed the homography matrix and calculated the approximation in terms of the angles of the pan/tilt platform on both cameras and the fully calibrated camera parameters. After the calculation of the homography, obstacles are detected on the point correspondences based on the distance between the projected coordinate. Error analysis was also conducted in their work.

Williamson and Thorpe [47] combined two calibration methods to find the fixed ground plane homography. And they also applied an adjustment to the homography by using nonlinear optimization in order to minimize the residue error between the warped images.

Zhang et al presented three algorithms for obstacle detection [52]. The first algorithm employs a calibrated camera to derive a linear system whose solvability implies the absence of obstacles. The second algorithm does not require camera calibration and exploits the homography of the ground plane to derive a linear system relating corresponding image coordinates in two views. Similar to the first algorithm, inconsistency of this linear system signals the presence of obstacles. The third algorithm uses sequences of partially calibrated stereo pairs to estimate the equation of the ground plane and the height of obstacles. However, this approach can only find whether the feature points are on obstacles but not all image points over the image.

Lourakis and Orphanoudakis in [31] is the work closest to ours. They assume the ground is locally planar and more than half of the feature points shown in the images are from the ground. They first solve the point correspondence problem by searching on epipolar lines. With the point correspondence found, they solve every homography matrix by applying LMedS to the points to reject the non-ground points in every image pair. However, using LMedS is just theoretical in solving statistical problems but not in real-time control, because the number of mathematical operations needed is impossible to process in the normal workstations.

2.1.3 Regression

Regression analysis, i.e. the problem of fitting a model to noisy data, is a very important subfield in statistics. The traditional approach to regression analysis employs the least squares method (LSM), which is popular due to its low computational complexity. LSM involves the solution of a linear minimization problem, and achieves optimal performance if the underlying noise distribution is Gaussian with zero mean. However, in cases where the noise is not Gaussian, or in the presence of outliers, that is observations that deviate considerably from the model representing the rest of the observations, the LSM estimator becomes highly unreliable. One criterion for characterizing the tolerance of an estimator with respect to outliers is its breakdown point, which may be defined as the smallest amount of outlier contamination that may force the value of the estimate outside an arbitrary range. As an example, LSM has a breakdown point of 0%, because a single outlier may have a substantial impact on the resulting estimation.

The Least Median of Squares (LMedS) estimator was originally proposed by Rousseeuw [37], which can handle data sets containing many outliers. LMedS involves the solution of a nonlinear minimization problem that aims at estimating a set of model parameters that best fit the majority of the observations. In contrast, LSM tries to estimate a set of model parameters that best fit all the observations. Thus, LMedS has a breakdown point of 50%, a characteristic that makes it particularly attractive for regression. However, on the other hand, it also requires tremendous number of calculations. If we want to reject 10% of outliers from a set of correspondence numbered 100, it needs about ${}_{100}P_{90}$=6.28e+19 calculations to find the suitable solution. For a real-time control operation, it is impossible to accomplish with common workstations as well as the fastest PCs.

2.2 Control Strategy

A controller is essential for a close loop robotic system. And the main usage of this controller is to work with our obstacle detection algorithm and completing the control loop. To test the performance and prove the possibility to implement in real-time process, goal-seeking behavior is not implemented in this system. For convenience and easy implementation, fuzzy controller is chosen as the controller of our robot. In this section, some work on navigation control strategy with fuzzy controllers are reviewed. e.g. [39], [2], [30] and [10].

Since the main contribution of this work is about generating an obstacle map over a legged motion from image data. For simplicity, we use a simple fuzzy control algorithm to make use of the obstacle map so acquired for commanding the legged motion. However, more sophisticated control strategies like the one in [39] can be used with our obstacle map as the input. The controller proposed in [39] by Saffotti et al, converts the range sensors inputs to vectors with different length pointed from the robot. By summing up the vectors and the desired goal direction with different weights, the control vector is found. With different weight sets, different complex behaviors can be simulated. Simulations of the robot trajectories are successful without collisions in narrow tunnels.

In [2], Barret et al designed a neural-fuzzy controller. Other than approaching to a specific goal, they implemented two navigation behaviors: the centering behavior and the left/right wall following behavior. The centering behavior, namely to maintain the robot at the center of the obstacles in order to keep away from the obstacles at both sides. The wall following behavior is implemented to avoid trapping in concave environments, such as withdrawing in dead-ends. In the implementation of the centering behavior, they firstly normalize the sensor inputs from three directions, left, right and front. The angular velocity is controlled by the first fuzzy controller using the normalized left and right inputs. Finally, the forward speed is controlled by using another fuzzy controller with the angular velocity and the normalized front sensor input.

In [30], Lian designed another navigation fuzzy controller. Like other researches, three sensor inputs are detected in left, right and front directions. He departed from using a multi-input fuzzy system, while he used three single-input fuzzy systems. Each subsystem determines its own commands to each motor. Finally, three sets of commands are added together to control the robot. In his work, the robot is successful in avoiding obstacles and turning around at dead-ends.

In [10], Chee et al designed a two-layered fuzzy controller. The first layer integrates all the sensor inputs to two outputs, the left and right clearance. The extracted information and the goal direction, total three inputs, are fed into the second layer fuzzy system.

In [43], Song and Sheen presented a neural fuzzy system. In principle, the algorithm likes the one presented by Chee et al in [10]. Firstly, 9 kinds of environment are predefined. Then, they use the neural network to classify the incoming sensor readings in real-time. The results of the classification are then put in the fuzzy controller and two motor commands are determined.

Although all the reviewed researchers built successful systems, what they implemented on are all wheeled robots. In this thesis, fuzzy controller is applied in controlling a legged robot that is seldom seen in the field of navigation control.

Chapter 3

System Overview

In this thesis, we exploit the concept of the homography for separating the ground and the obstacles in a pair of images. We take advantage of the fact that obstacles are always not lying on the ground plane and thus their projections in the images would not agree with the homography the ground induces. Fig.2 shows the flow chart of the navigation system.

With the cameras facing toward the front, and assuming that obstacle detection and avoidance in the preceding moments has run effectively, at any particular time instant the features at the very bottom of the stereo images would consist of features projected from the ground. With a few feature correspondences over those areas available, which can often be extracted using the epipolar constraint [26] or the quasi-invariance of the features' attributed characteristics, the homography induced by the ground across the two images can be estimated. The homography allows the right image to be warped to the left view. The original left image and the warped right image would be consistent over features from the ground, but not over features not from the ground. Information about the obstacles can thus be extracted on the left view, and such a 2D obstacle distribution can be warped to the ground through another homography, which in turn allows obstacle information on the ground be extracted in terms of a 2D map we refer to as the Local Obstacle Map (LOM). The essence of the approach is that an explicit 3D reconstruction of the surroundings is no longer necessary.

A simple and classical fuzzy controller can then be used to command the robot's global motion, the speed and steering, based on the LOM information. By implementing the suitable rules to the rule table, the robot can avoid large obstacle, such as walls, and small hills. Based on different kind of environment, it will behave to turn left, turn right and reverse.

As we do allow the robot motion and the LOM estimation to take place at the same time, the time delay of the LOM information ought to be addressed. Consequently, we have included a prediction mechanism to compensate the LOM for the upcoming time instant from the time-delayed input images.

A complete navigation system has been implemented and tested with real environments. It consists of a six-legged robot of the size about 13"x12"x10", a stereo pair of black-and-white miniature cameras with wide angle lenses mounted on it, and a Pentium II remote-brain connected to via wireless video transmission. The remote brain is where most of the calculations take place.

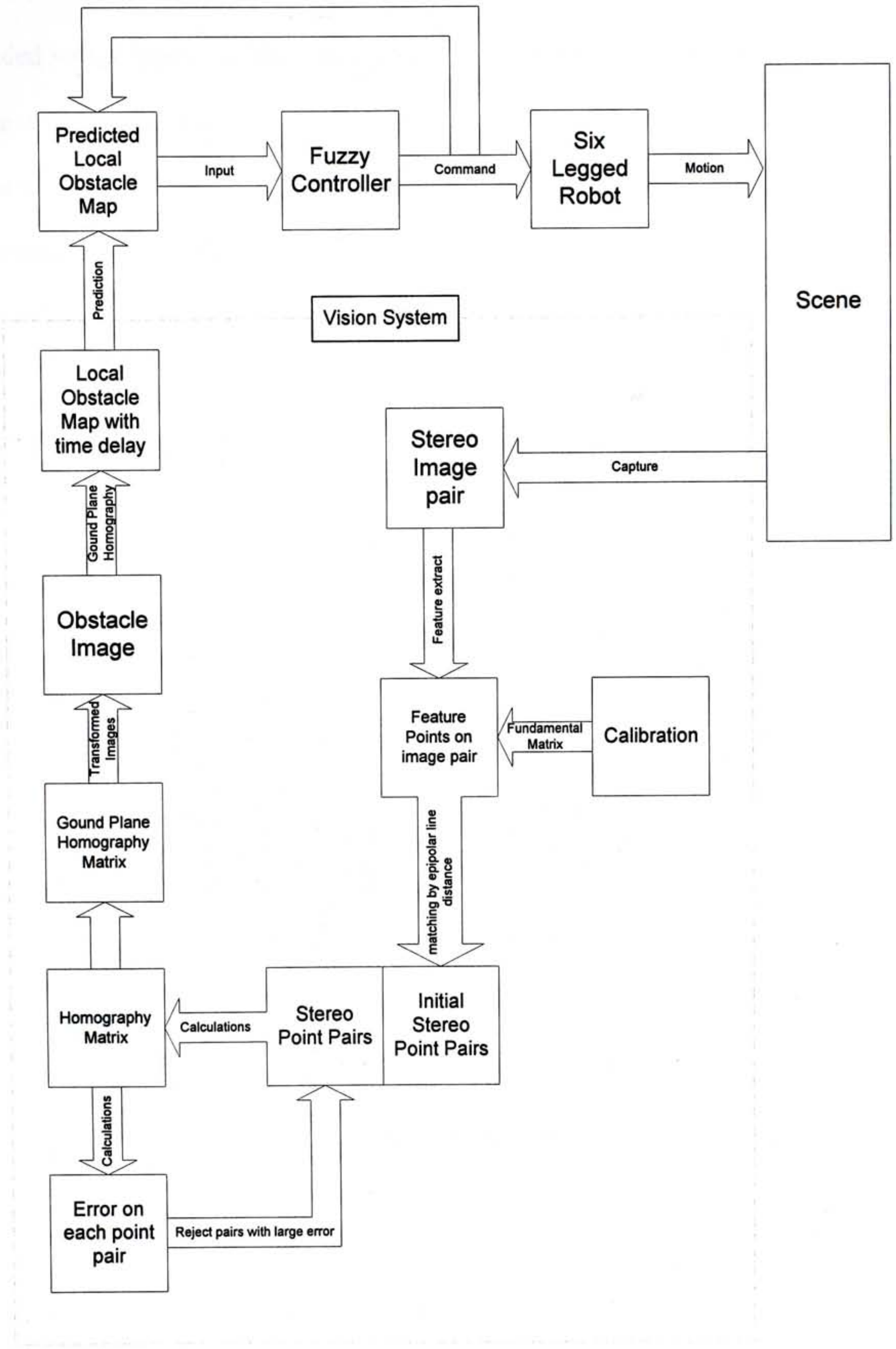


Fig.2 System overview flow diagram

Fig.3 shows the robot. The robot has each leg activated by two RC servos, and is embedded with a Motorola 68332 microcontroller that drives all the servos. Experiments in textured environments show that the system is effective for autonomously and real-time navigating along straight paths and curved corridors while avoiding walls on the two sides, making turns at corners, and withdrawing from dead-ends with image resolution of 384 by 288.

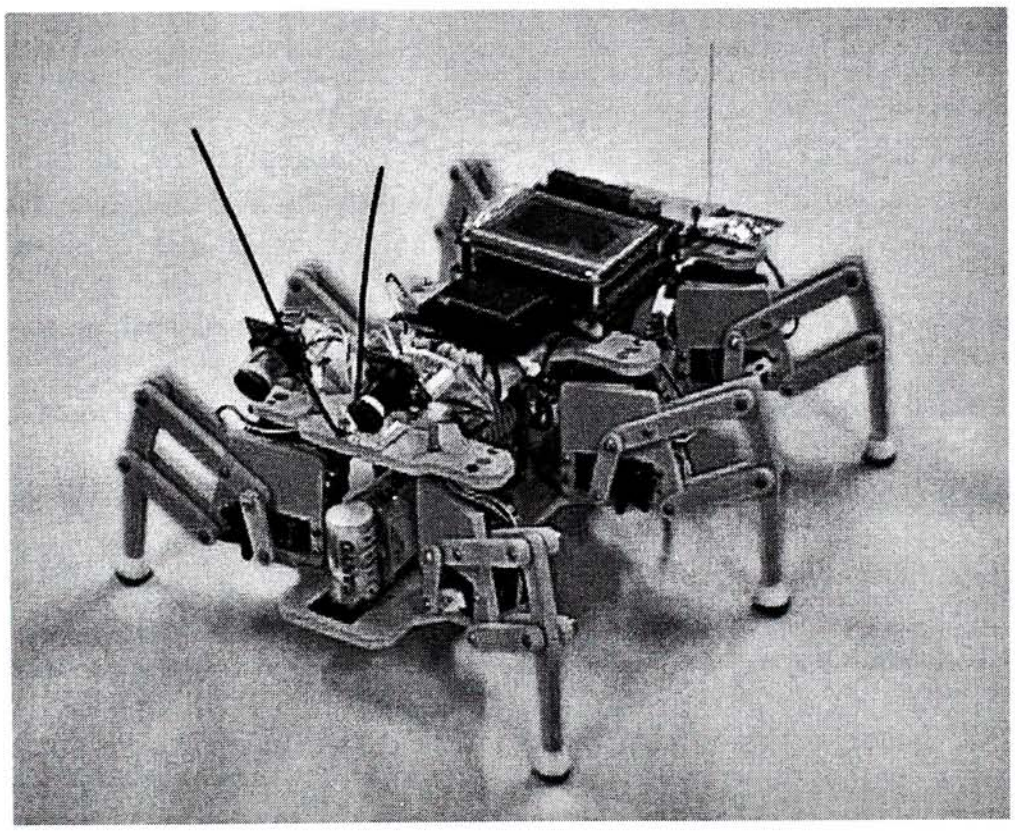


Fig.3 The six-legged mobile robot used.

Chapter 4

Obstacle Detection by Fast Homography Estimation

Homography is a mapping across two images induced by a plane in 3D. It is captured by a 3 by 3 matrix *H*, a characteristic of the plane, such that any point pair projected by that plane to the two images satisfy $m' \cong H\,m$, where $(m\,,\ m')$ are the homogeneous coordinates of the two image positions, and $\cong$ denotes equality up to a scale. Features that are not on that plane generally do not satisfy the mapping. A homography can be estimated from as few as 4 pairs of feature points projected from the associated plane.

In this chapter, we describe how we detect obstacles from the property that their image projections do not satisfy the homography the ground induces. We shall describe how image features projected from the ground are obtained, how they are matched, and how such matches are used to estimate the homography induced by the ground. With such a homography that we refer to as the ground homography available, the obstacles can be detected. Naturally, we require that the environment including the ground has sufficient features to be detected and matched across the images.

4.1 Ground Feature Extraction

We assume that at any particular time instant the obstacle detection and avoidance in the preceding instants has run effectively. As a result, at any instant the immediate front of the robot is free of obstacles. With this, the features at the very bottom of the stereo images would consist only of features projected from the ground. We detect feature points in those image areas using a simplified method [4] from the filter-based method proposed by Tomasi and Kanade in [44]. It finds the features by examining the intensity gradients in two orthogonal directions. If at any image position both gradients are large, we regard the image position as a textured feature point. In Fig.4, the upper image pair shows the textured environment. The lower images show the detected featured points in small white "+" sign.

4.2 Ground Feature Correspondence

We have the epipolar geometry of the two cameras estimated in terms of a fundamental matrix *F*, in an offline process using a method similar to the one in [51]. With the fundamental matrix available, for any image position m_i (in terms of homogeneous coordinates) in one view the locus of its correspondence in the other view is known as a line named as the epipolar line. The epipolar geometry with the epipolar plane and lines are shown in Fig.5.

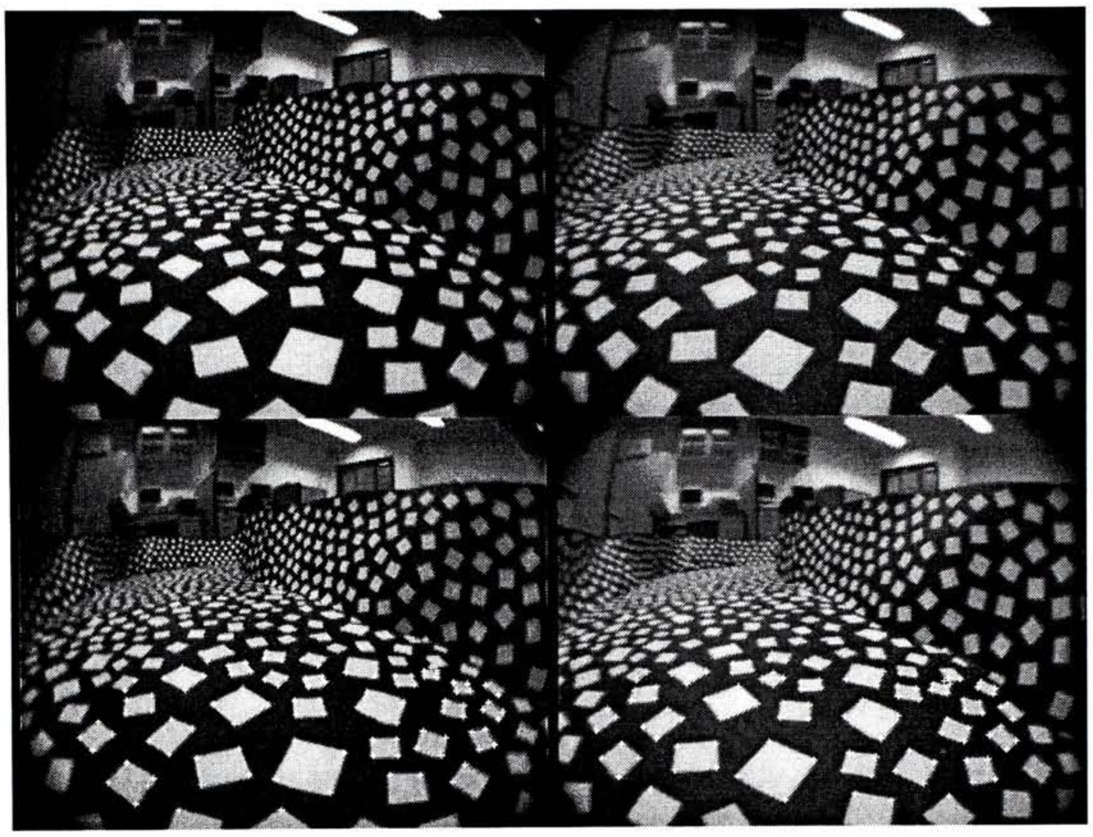


Fig.4 Detected feature points on an image pair

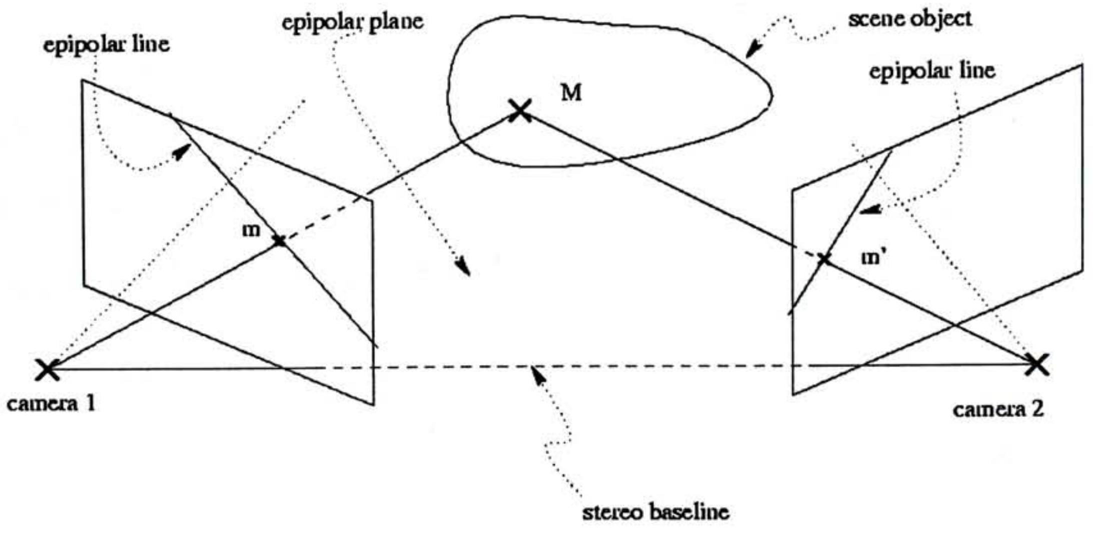


Fig.5 Stereo geometry and epipolar lines

We extract a number of feature correspondences over the ground using this epipolar constraint and the information of the previously estimated homography matrix in order to minimize the search area. For every pair of feature points in the two views extracted above, we define their correspondences' error against the epipolar constraint as the normal distance to the corresponding epipolar line:

$$E_{i,j} = \frac{\left| m_j'^T F m_i \right|}{\sqrt{\widetilde{m}_{i,1}^{\,2} + \widetilde{m}_{i,2}^{\,2}}} \tag{1}$$

where m_i is the i^{th} feature point in the left image in homogenous coordinates, m_j' is the j^{th} feature point in the right image also in homogeneous coordinates, and $\widetilde{m}_{i,1}$ and $\widetilde{m}_{i,2}$ are such that $\begin{bmatrix}\widetilde{m}_{i,1} & \widetilde{m}_{i,2} & \widetilde{m}_{i,3}\end{bmatrix}^T = Fm_i$.

We take it that m_i and m_j' are a correct correspondence pair if their error is small enough (2 pixels in our implementation). At this stage, we do not insist that one point is only matched to one point in the other view. All the pairs of feature points so extracted will be the initial input for the estimation of the ground homography.

4.3 Ground Homography Estimation

For a fast implementation we use only a linear method, whose details are covered in section 4.3.2, to estimate the homography. However, linear methods generally suffer from more errors, and one important source of such errors is the uneven distribution of the input points, which causes some input points to dominate over the others in the estimation process. It is therefore necessary to transform the input points to give them equal voting power before supplying them to the estimation process.

4.3.1 Input point transformation

Here we aim at transforming the homogeneous coordinates of the feature points on the two views respectively to two separate sets of homogeneous coordinates, each of which close to a unit spherical surface and spanning the surface as evenly as possible:

$$\sum_{i=1}^{n} \hat{m}_i \hat{m}_i^T = nI \qquad \sum_{i=1}^{n} \hat{m}'_i \hat{m}'^{\,T}_i = nI \tag{2}$$

where n is the total number of point pairs, I is the 3 by 3 identity matrix, $\hat{m}_i$ and $\hat{m}'_i$ are the 3 by 1 vectors representing the transformed homogenous coordinates of the ith point pair on the two views.

The idea is adapted from the method proposed in [22], there was a fundamental matrix to be estimated but here is a homography. We first find two 3 by 3 matrices A and A' from the point pair coordinates:

$$A = \sum m_i m_i^T \qquad A' = \sum m_i' m_i'^T \tag{3}$$

Using Cholesky Factorization, we can decompose A and A' into the lower triangular matrices L and L':

$$nA = LL^T \qquad nA' = L'L'^T \tag{4}$$

Then the required transformation is merely

$$\hat{m}_i = L^{-1} m_i \qquad \hat{m}_i' = L'^{-1} m_i' \tag{5}$$

4.3.2 Initial estimation

The set of equations $\hat{m}_i' = \hat{H}\hat{m}_i$, $\forall i \in \{1,2,\cdots,n\}$ can be expressed as

$$\overbrace{\begin{bmatrix} \hat{m}_{i,1}\hat{m}'_{i,3} & \hat{m}_{i,2}\hat{m}'_{i,3} & \hat{m}_{i,3}\hat{m}'_{i,3} & 0 & 0 & 0 & \hat{m}_{i,1}\hat{m}'_{i,1} & \hat{m}_{i,2}\hat{m}'_{i,1} & \hat{m}_{i,3}\hat{m}'_{i,1} \\ 0 & 0 & 0 & \hat{m}_{i,1}\hat{m}'_{i,3} & \hat{m}_{i,2}\hat{m}'_{i,3} & \hat{m}_{i,3}\hat{m}'_{i,3} & \hat{m}_{i,1}\hat{m}'_{i,2} & \hat{m}_{i,2}\hat{m}'_{i,2} & \hat{m}_{i,3}\hat{m}'_{i,2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}}^{\hat{M}} \bullet vec\left(\hat{H}\right) = 0 \qquad (6)$$

where $\hat{H}$ is the homography associated with the transformed point coordinates, and is defined up to an arbitrary scale. Without loss of generality, we can set the Frobenius norm of $\hat{H}$ to 1, i.e., $\|\hat{H}\| = 1$. The least-square solution of $\hat{H}$ is then the unit eigenvector associated with the least eigenvalue of the matrix ($\hat{M}^T\hat{M}$). However, in our implementation to speed up the process we take one it its entries as 1 and estimate $\hat{H}$ using a simple matrix inversion process. In the case that some of the other entries of $\hat{H}$ obtained very large magnitudes, which indicate that the entry assumed one is close to zero compared to them, we fixed one entry of $\hat{H}$ as 1 and get another estimation, and this process goes on until we get an $\hat{H}$ of finite entries.

Since $\hat{m}'_i = \hat{H}\hat{m}_i$ and $m'_i = L'\hat{m}'_i = L'\hat{H}L^{-1}m_i$, the homography H associated with the original point coordinates is:

$$H = L'\hat{H}L^{-1} \qquad (7)$$

4.3.3 Robust estimation

There can be wrong matches or matches not coming from the ground in the above feature points, and they cause error in estimating the ground homography. Assuming most of the input feature points are correct matches from the ground, we use a robust estimation process to remove the outliers and improve the estimation accuracy. Fig.6 shows the process flow diagram we use.

The error of the i^{th} point pair is defined as the Euclidean distance between one point and the position on the same view projected from the other point using the homography:

$$E_i = \sqrt{(m'_{i,1} - \frac{H_1 m_i}{H_3 m_i})^2 + (m'_{i,2} - \frac{H_2 m_i}{H_3 m_i})^2} \qquad (8)$$

The error threshold we use in our implementation is 1.6

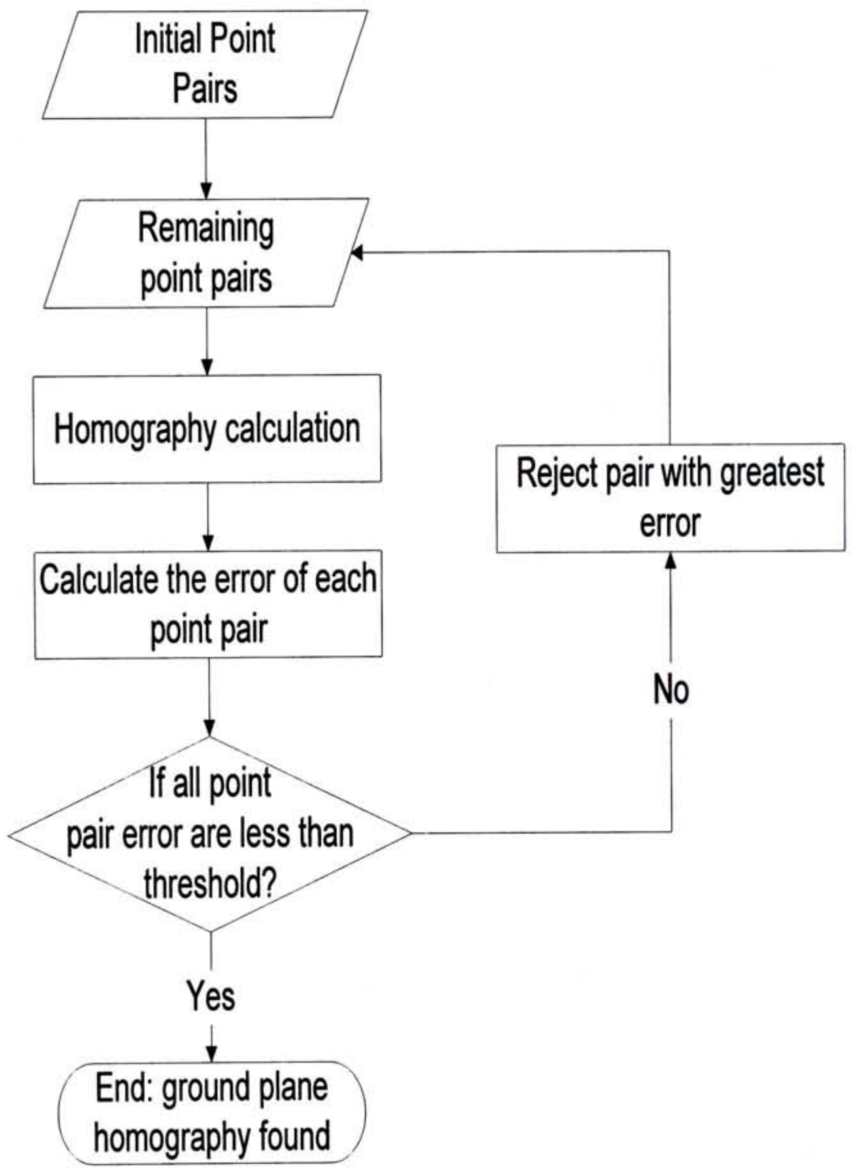

Initial Point Pairs
Remaining point pairs
Homography calculation
Calculate the error of each point pair
If all point pair error are less than threshold?
No
Reject pair with greatest error
Yes
End: ground plane homography found

Fig.6 Recursive homography calculation

4.4 Obstacle Detection

Once we have the ground homography H, we can use it to project one image to the other image, and the obstacles can be detected as the places where the first image and the warped image do not agree. Using the equation $\tilde{m}_i' \cong Hm_i$ or $\tilde{m}_i \cong H^{-1}m_i'$ to find the left warped to right image or the right warped to left image.

Fig.7 shows a stereo pair of regulated images of one of our testing environments, and Fig.8 shows the right image warped to the left view. In Fig.9, the magnitudes of the differences of the images are shown. It can be seen that feature points on the ground have almost zero values whereas feature points not from the ground like those from the walls score higher. The difference image over the view indicates the obstacle distribution in an image space.

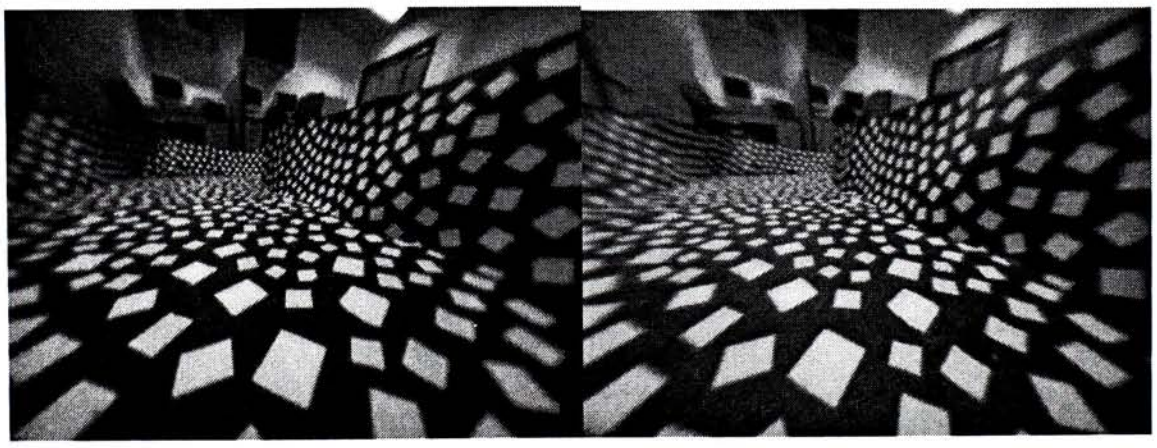


Fig.7 Perspectively projected image pair

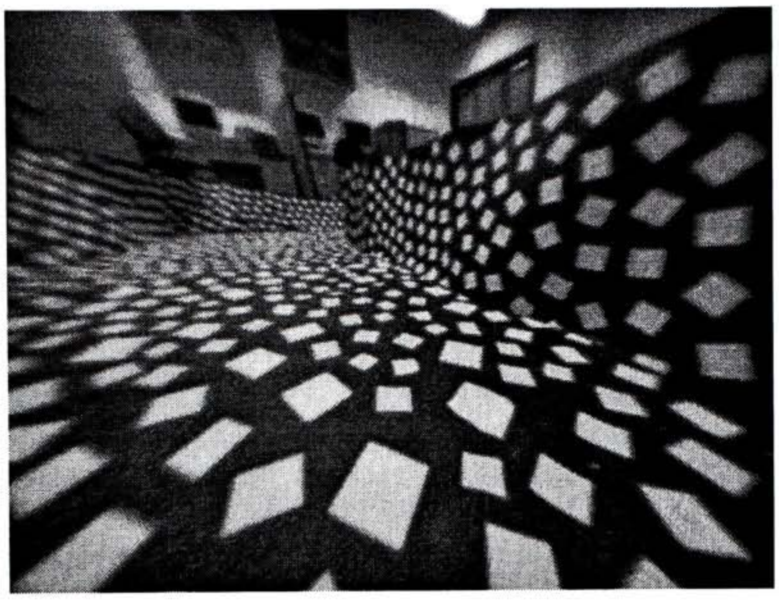


Fig.8 Right image warped to the left view

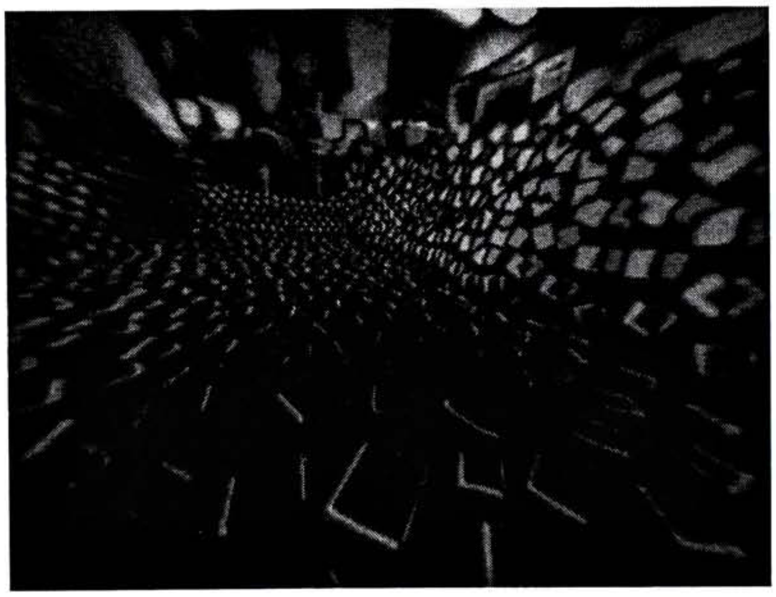


Fig.9 Difference of the two images

Imperfections in the difference image come from the inaccuracy of the ground homography, the unevenness of the ground, the different illuminations of the two original images and their finite resolutions. To enhance the contrast between the obstacles and the ground features in the difference image, we apply a simple filtering process to it.

We calculate the sum of the differences over a small window (of size 20x15 in our implementation) around every point (*i, j*) and see if it is bigger than a threshold (2000 in our implementation). If the magnitude of the sum is larger than the threshold, the filtered region is regarded as an obstacle.

Feature point pairs detected, which do not satisfy the ground homography, can be features from the near obstacles, from the ceiling (if the environment is an indoor one), or from the obstacles very far away. Since for robot navigation it is the near obstacles that are of the immediate concern, we would only regard near obstacles as the obstacles, and they are taken as the areas around the boundary of the ground area in the left view. Fig.10 shows the detected obstacles in a left view.

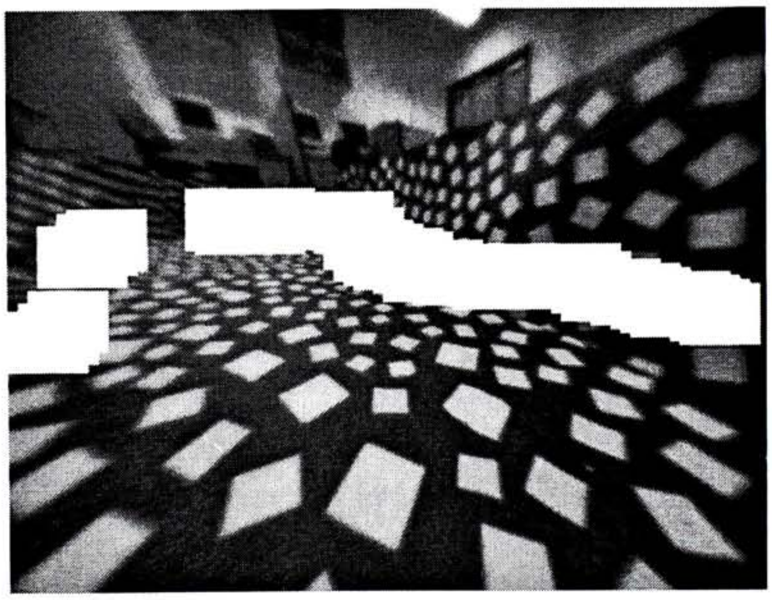


Fig.10 Obstacles detected in left view

The left view and the ground surface are yet another pair of planes related by a homography that is induced by the ground. We pre-estimate this homography in an offline process, and assuming it be approximately constant over time, use it to warp the obstacle information from the right view to construct the 2D obstacle distribution on the ground. (See Fig.11)

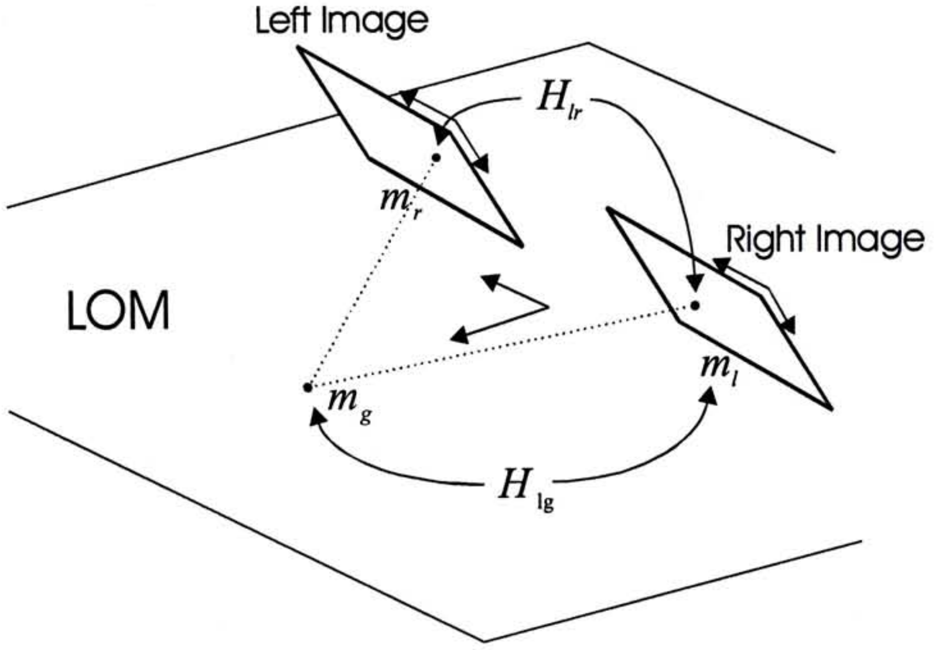


Fig.11 Homography projections

While an assumed constancy of the homography between the stereo views would cause failure in detecting the obstacles, here the assumed constancy of the homography between the left view and the ground would only distort the detected positions of those obstacles on the ground. The distortions are found to be minimal and tolerable in our implementation.

The advantages of this obstacle detection mechanism include that only the fundamental matrix of the stereo setup but not the camera calibrations is needed, no explicit 3D reconstruction of the surroundings is necessary, and that all the steps are simple enough to be implemented fast.

We do not make the claim that such obstacle information is without errors. Once in a while isolated error from various sources can occur at a particular time instant. In the next subsection we shall see how the obstacle information can be further improved by accumulating the information over a period of time and how its time delay can be compensated.

4.5 Local Obstacle Map (LOM) on Ground

The local obstacle map (LOM) in our system is a 41 by 41 array over the ground, as shown in Fig.12, with each array element having a value ranging from 0 to 1 which indicates how probable it is to have an obstacle responsible for an area on the ground of the size about 10cm by 10cm at the corresponding position on the ground. The value 1 at any particular entry indicates that the corresponding position on the ground definitely contains an obstacle, whereas the value 0 indicates the opposite. In practice, when we need to make a decision about whether there is an obstacle or not in a particular position, we take it that any value larger than 0.5 means a yes, and any value smaller than 0.5 means a no. A similar probability map has been used in [8], but it is used there for object localization rather than robot navigation. Here without any information about the obstacles at the very beginning, all the entries of the LOM are initialized with the value 0.5.

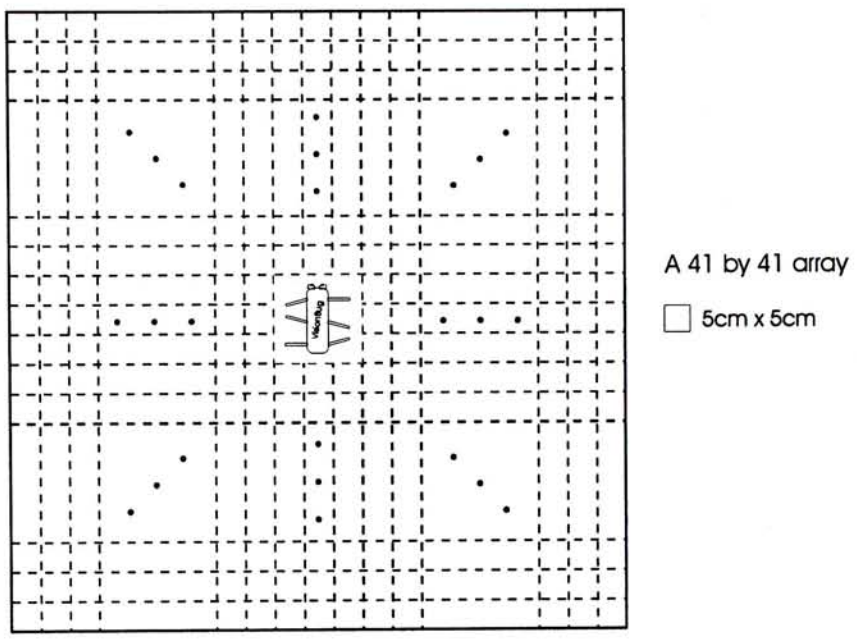


Fig.12 Design of Local Obstacle Maps

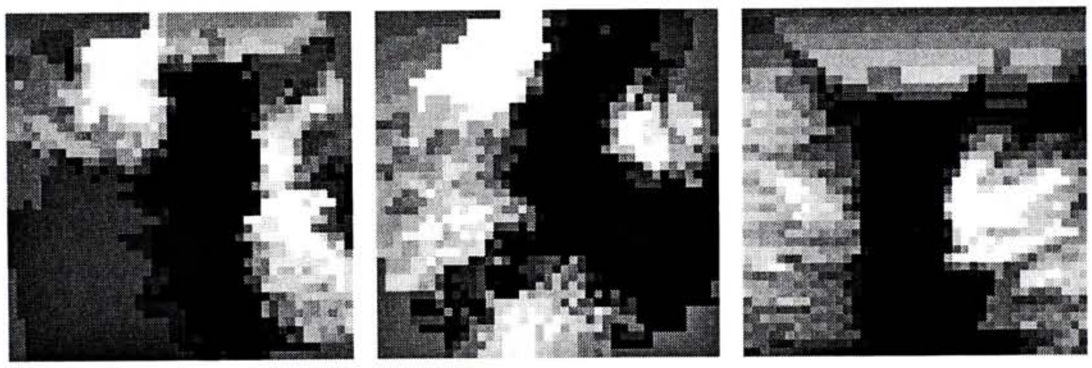
4.5.1 Extraction from accumulative evidence

At every time instant the stereo images would allow an obstacle distribution on the ground to be acquired. If obstacle is detected at a particular position of the ground, the corresponding entry of the LOM will be increased by 0.1. If obstacle is not detected, the corresponding entry of the LOM will be decreased by 0.1. This way the LOM is constructed based on accumulative evidence over a longer period of time, and is less prone to mistakes made at any particular time instant.

4.5.2 Time-delay compensation

We allow the robot to execute motion while obstacles are detected. This would cause the LOM to be obsolete by the time it is estimated. We compensate the time-delay by predicting the robot translation and rotation from the motion commands given to the robot while the last image pair is taken, and using such a robot motion to remedy the LOM. There would be errors in the compensation, as the robot motion is just a prediction, but since the LOM is updated from the most current image pair every time they are available, such errors are not accumulative and are tolerable.

Fig.13 shows a few LOMs for different environments. Fig.13 (a) is the case where the robot is walking between two walls. Fig.13 (b) is the case where there is a road leading to the right. Fig.13 (c) is the case where there is a road cornering to the right. All the LOMs were captured from a real robot exploration in real-time.



(a) straight path **(b) right going path** **(c) right angle path**

Fig.13 Sample Local Obstacle Maps (White blocks indicates obstacles)

Chapter 5

Obstacle Avoidance by a Fuzzy Controller

A controller is essential for a close loop robotic system. And the main usage of this controller is to work with our obstacle detection algorithm to test the performance and prove the possibility to implement in real-time process.

Application of fuzzy logic or neural-fuzzy in robot navigation is not a new topic, e.g. [43][10], but most of them aimed at using sonar or IR range sensors to detect distance to the obstacle. In [43], an example of navigation control with a neural-fuzzy controller was introduced. In their approach, obstacle distances in five directions are detected. By fuzzy logic, the shape of the environment is divided into a few common types. Each type of environment has its corresponding control command. A neural network is used in combining output commands from all possible commands.

In [10], a two-layered fuzzy controller is introduced. The first layer integrates all the sensors' information into two valuables, the left and right clearance. The other layer determines the control commands making use of the clearance valuables. The goal direction also encountered.

Since our robot is six-legged and already has a sub-controller to drive the leg motion. Based on one of the gait patterns covered in [3], we designed the leg motion sequence.

With the difference of the magnitude of the leg movement on two sides, we can control the forward and rotating speed of the robot. So what are left to control are only two degrees of freedom: the direction of steering and the walking speed in the global robot motion.

Fuzzy controller is one of the simple controllers that can handle rules without complicated implementation. A simple fuzzy controller with classical fuzzy logic is designed to achieve the task. It is a controller with 3 inputs and 2 outputs. Without a specific goal to reach, our robot is a behavioral robot with the following rules for its navigation:

- The robot walks among obstacles in a way that the obstacles are evenly distributed on the two sides of the robot.
- If there is no obstacle in the front, the robot walks directly to the front.
- If there are obstacles on the left, it steers toward the right, and vice versa.
- The walking speed of the robot decreases with decreasing distance to the obstacles.

What we do is to translate these navigating behaviors into linguistic rules to the rule table of the fuzzy controller.

5.1 Gait Pattern

According to the research of Beer et al [3], the motion of an individual leg from a multi-legged animal, including all kinds of animal, can be separated into two phases, the stance phase and the swing phase. As a legged creature moves towards the forward direction, in order to push the body to the front, it always presses its legs on the ground and moves from the front to the back with respect to the body. In the period the legs moving backward is called the stance phase. Then since limitations exist at all joints, the leg would be bounded. At the moment the leg reached the limit at back most, the leg will reactively pick up and move to the most forward position very quickly. The period the leg suddenly moving to the front is called the swing phase. Namely, in the swing phase, the leg moves like a pendulum and swing to the front, the energy consumed is as little as possible. When these two phases changes recursively, the body moves to the forward direction. Moreover, these two phases are activated by neurons, which form a small neural control system. Each leg individually has its own neural control system and the structure of the networks is created when the legged creature is built or born.

Since the motion of a leg is performed recursively, when the phase cycles of the legs are being activated is the key problem to move the body globally and effectively. To coordinate all the legs at the same time seems to be a difficult problem. Insects researchers found that all insect gaits are characterized by a stepping sequence known as a metachronal wave in which sequences of swings propagate from the back of the insect to the front on each side of body. Based on this control structure, multi-legged creatures, such as worms, although have more than 10 pairs of legs, they are still controllable with a periodical control signal without disorder or mess-up with the legs.

In addition, by the concept of metachronal wave, only two degrees of freedom remain to control the global motion of a legged creature. If the frequency of the metachronal wave of both sides increased, the speed of the global motion in the forward direction will also increased. If the frequency of the metachronal wave in the left side is higher than the right side, the global motion of the body will probably turning to the right and vice versa.

Applying the same concept in robot leg control, the six legs on the robot in our experiment are divided into three sets. The path of the motion of a leg is shown in Fig.15. In the figure, it shows the path of this contact point, at where the leg contacts the ground (Fig.14). It also shows that the motion of a single leg is divided into 6 states. The controlled points and the actual motion are represented by dotted lines and a solid curve respectively.

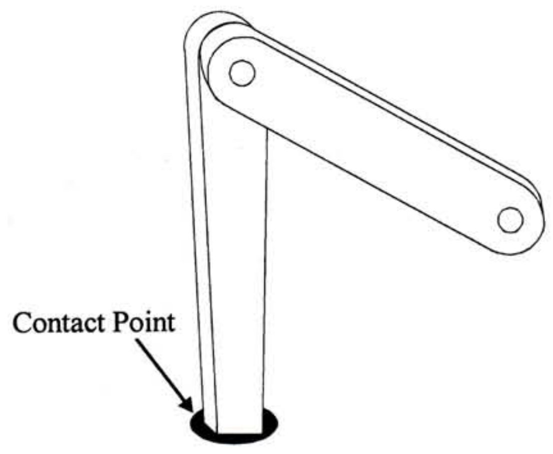


Fig.14 Contact point of a leg

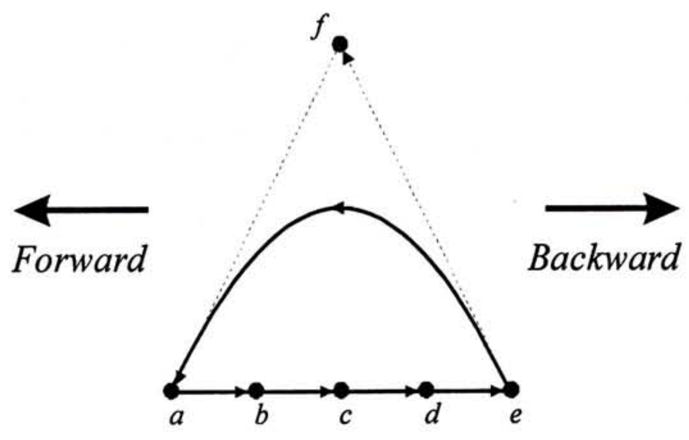


Fig.15 Side viewed motion of a leg

At the same moment, the legs in the same set are commanded to position at the same state. Considering supporting the robot evenly, the sets of legs are organized as follows:

Set 1:	Left Front & Right Middle
Set 2:	Left Middle & Right Rear
Set 3:	Left Rear & Right Front

The arrangement and the leg position are shown in Table.1. In the fourth row of the table, small pictures representing the states are also shown. A black dot at the end of a leg indicates contact between the leg and the ground. If no dot on the leg indicates that the leg is picked up and stepping forward in the swing phase.

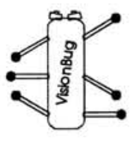
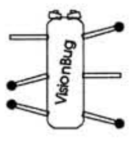
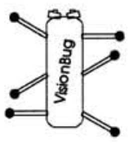
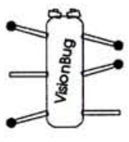
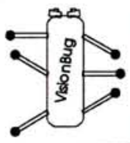
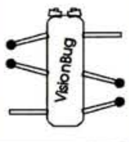
	State 1	State 2	State 3	State 4	State 5	State 6
Set 1	e	f	a	b	c	d
Set 2	c	d	e	f	a	b
Set 3	a	b	c	d	e	f
						

Table.1 Legs configurations in each state

From this leg configuration, we can see that all of the time the robot walks with four legs supporting the body. This configuration can effectively alleviate the vibrative walking motion.

5.2 Fuzzy Logic Controller

5.2.1 Controller Inputs

For simplicity, we only extract three inputs from the local map. It is because 3 is minimum number of inputs that can handle most of the sceneries. If one more input is used, the number of rules will be increased from 27 to 64. Moreover, it is not easy to define so much rules manually.

The inputs are defined by the shortest distance of the obstacles to the robot in the left, front and right directions (i_L, i_F and i_R). In practical, these values are measured from the predicted LOM, as shown in Fig.16. These variables are then fuzzfied to linguistic variable sets I_L, I_F and I_R, which contains linguistic variables.

The cardinal of the linguistic variables set is chosen as 3, which will allow us to describe 27 different rules. The linguistic variables are denoted as:

S	:	Short
M	:	Medium
L	:	Long

The 3 inputs use the same membership function. The shape of the membership function is triangular and shown in the Fig.17

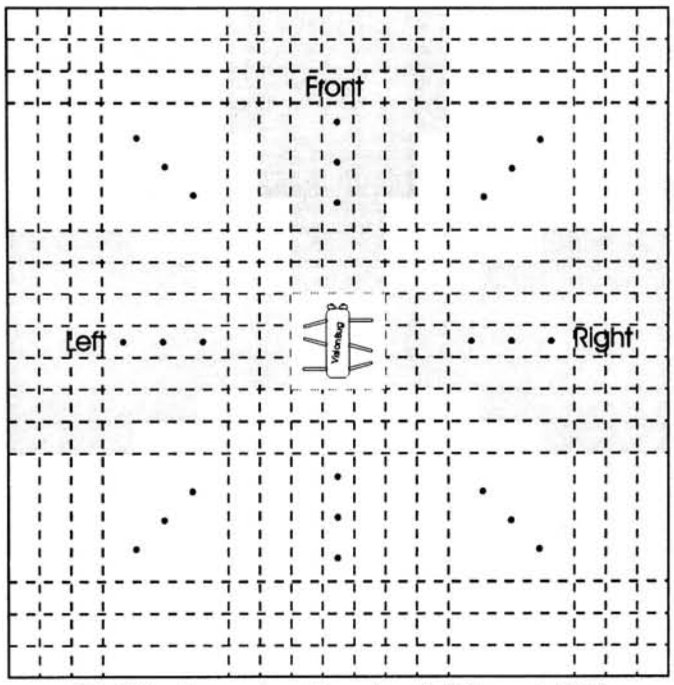


Fig.16 Fuzzy inputs extracted from LOM

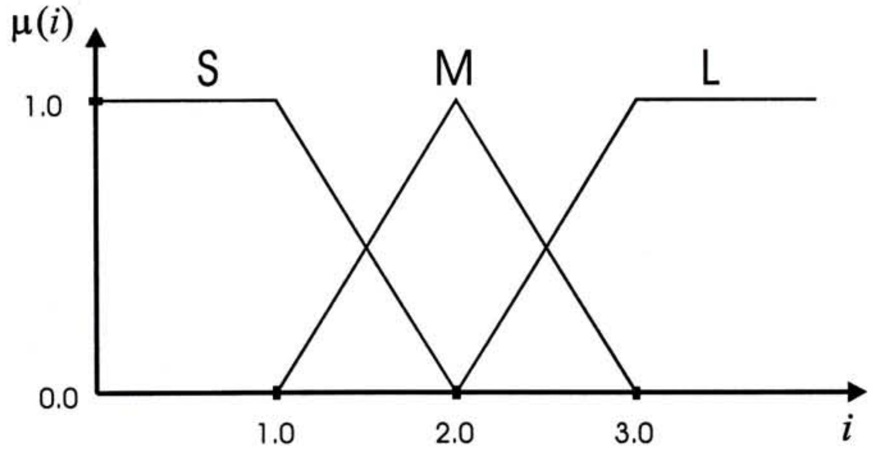


Fig.17 Input fuzzy membership function

5.2.2 Controller Outputs

Although the robot is six legged, it still walking on a 2D plane. Like other wheeled robots, we only need to command the robot in two variables, the steering and the speed.

According to these two variables, gait controller can assign the legs motion to accomplish the desired global motion in order to avoid obstacles.

Fig.18 and Fig.19 show the membership functions of the steering output and speed output respectively.

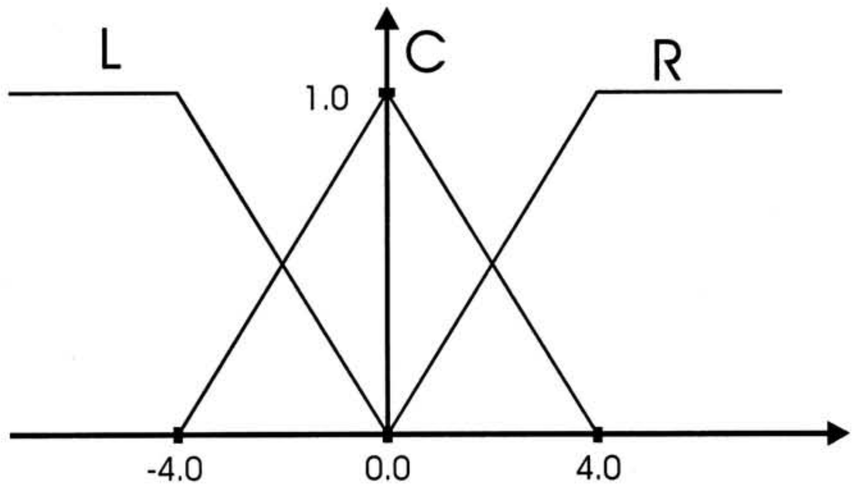


Fig.18 Steering output membership function

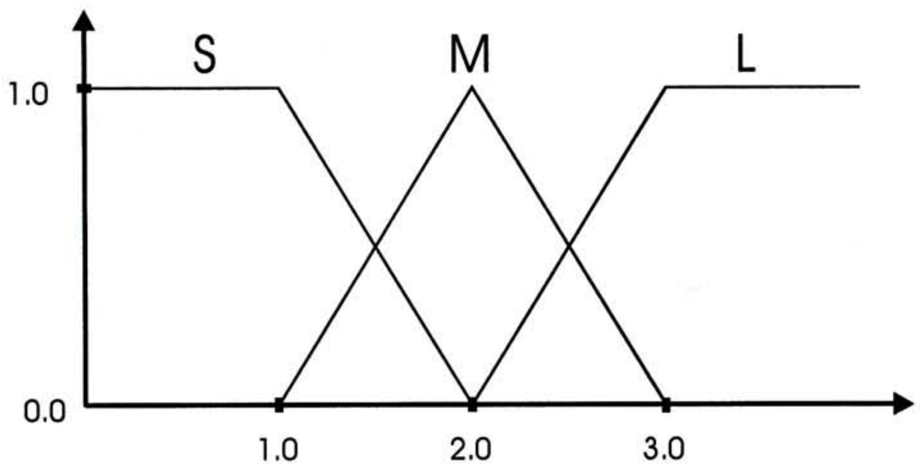


Fig.19 Speed output membership function

5.2.3 Inference mechanism

Fuzzy systems are based on the integration of expert knowledge through fuzzy reasoning. Let us define a general rule base form as

R1: if I_L is A1, I_F is B1 and I_R is C1, then O_{steer} is D1 and O_{speed} is E1

R2: if I_L is A2, I_F is B2 and I_R is C2, then O_{steer} is D2 and O_{speed} is E2

$$\vdots$$

where I_L, I_F and I_R are linguistic variables representing the process state variables. O_{steer} and O_{speed} are linguistic variables representing the control variables. Ai, Bi, Ci, Di and Ei are linguistic values of the linguistic variables with i=1, 2, ..., 27.

The rules can be described by two 3 by 3 by 3 matrices as shown in Fig.20. The linguistic values in the tables are chosen manually in real experiments that the robot can avoid obstacles. And these may not be the rules that perform the optimal performance.

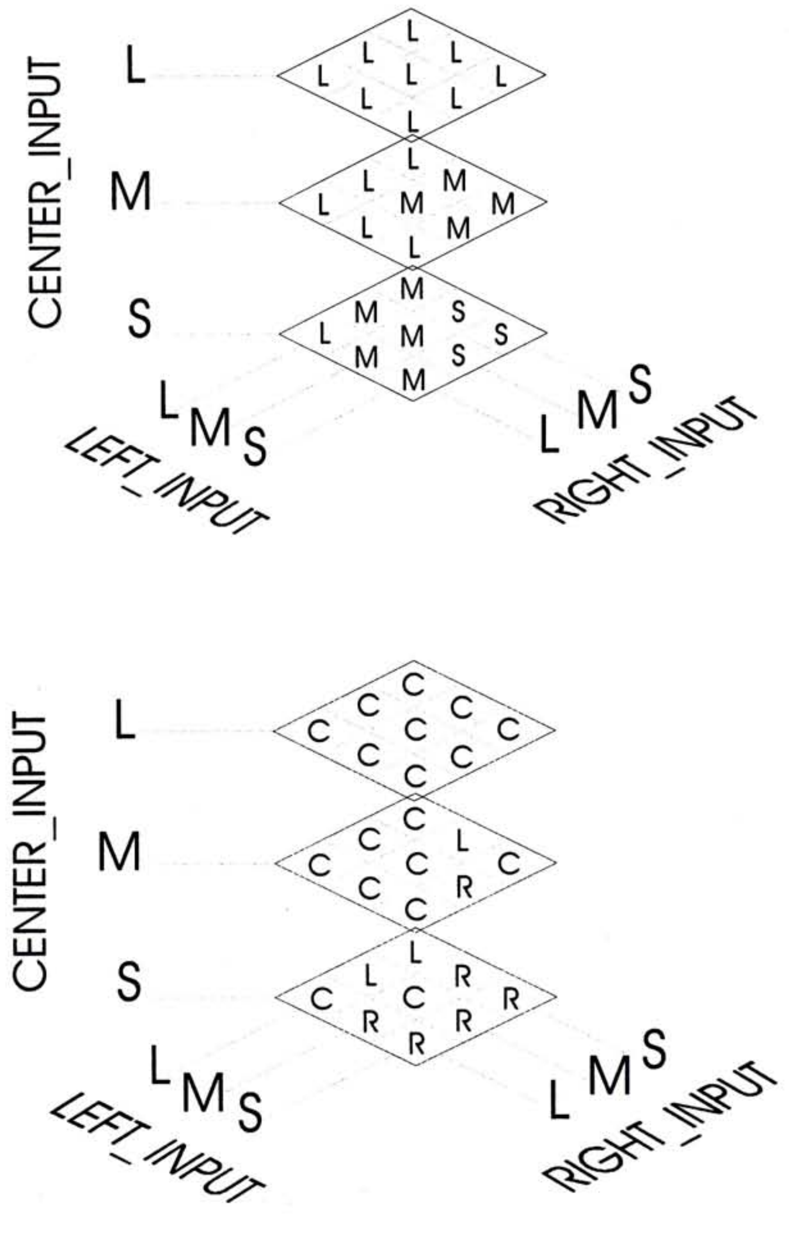


Fig.20 3D rule table for the speed output and the steering output

Let α_i be the measure of the contribution of the i^{th} rule to the fuzzy control action. For rules *i* we have:

$$\alpha_i = \mu_{A_i}(i_L)\mu_{B_i}(i_F)\mu_{C_i}(i_R)$$

and leads to the control decisions:

$$O_{steer} = \sum_{i=1}^{27} \alpha_i d_i \qquad O_{speed} = \sum_{i=1}^{27} \alpha_i e_i$$

where d_i and e_i are the actual values of the linguistic control values Di and Ei which defined in the output membership functions.

Chapter 6

Implementation

In this chapter, hardware specification and the capabilities will be presented and focus on the vision side, since it is a computer vision application. The calibration of the video camera and the motion of the robot will also be included.

6.1 Hardware components

Hardware is one of the important contributions in this thesis. Other than a legged robot—VisionBug, there are still other components to make the robot controlled remotely by a personal computer only based on the captured visual data. This section describes the hardware, the connections and the signal flow between devices.

In Fig.21, the diagram shows the major hardware components and their connections in this remote-brain control system. There are two main components, the robot and the PC.

6.1.1 VisionBug

VisionBug namely is a bug like robot small robot that use computer vision in robot navigation. It is also the hardware platform we implement our developed algorithms.

VisionBug is a legged small mobile robot converted from RugWalker, which is purchased from Joker Robotics. It is 13" long and 11" wide with stretched legs. The height is about 10" when it walks and it weight about 1.5kg including batteries.

On the robot, other than the leg motors, a micro-controller board, a serial signal receiver, two circuit board cameras and two miniature video transmitter are attached. The receiver board connected to the serial port of the micro-controller board and the twelve motors connected to the PWM ports. Although the video cameras and transmitters are attached on the robot, since they are used for capture images from the robot and connected wirelessly to the PC, no connections are needed between the micro-controller boards.

The PC just like other common desktop computers but installed a video grabber – Matrox Genesis, which is a high-end grabber with processors and 64Mbyte memory onboard. This PC is also where most of the calculation-extensive jobs take place.

To complete the control loop, other individual hardware such as the TV tuners, serial signal transmitter board are used.

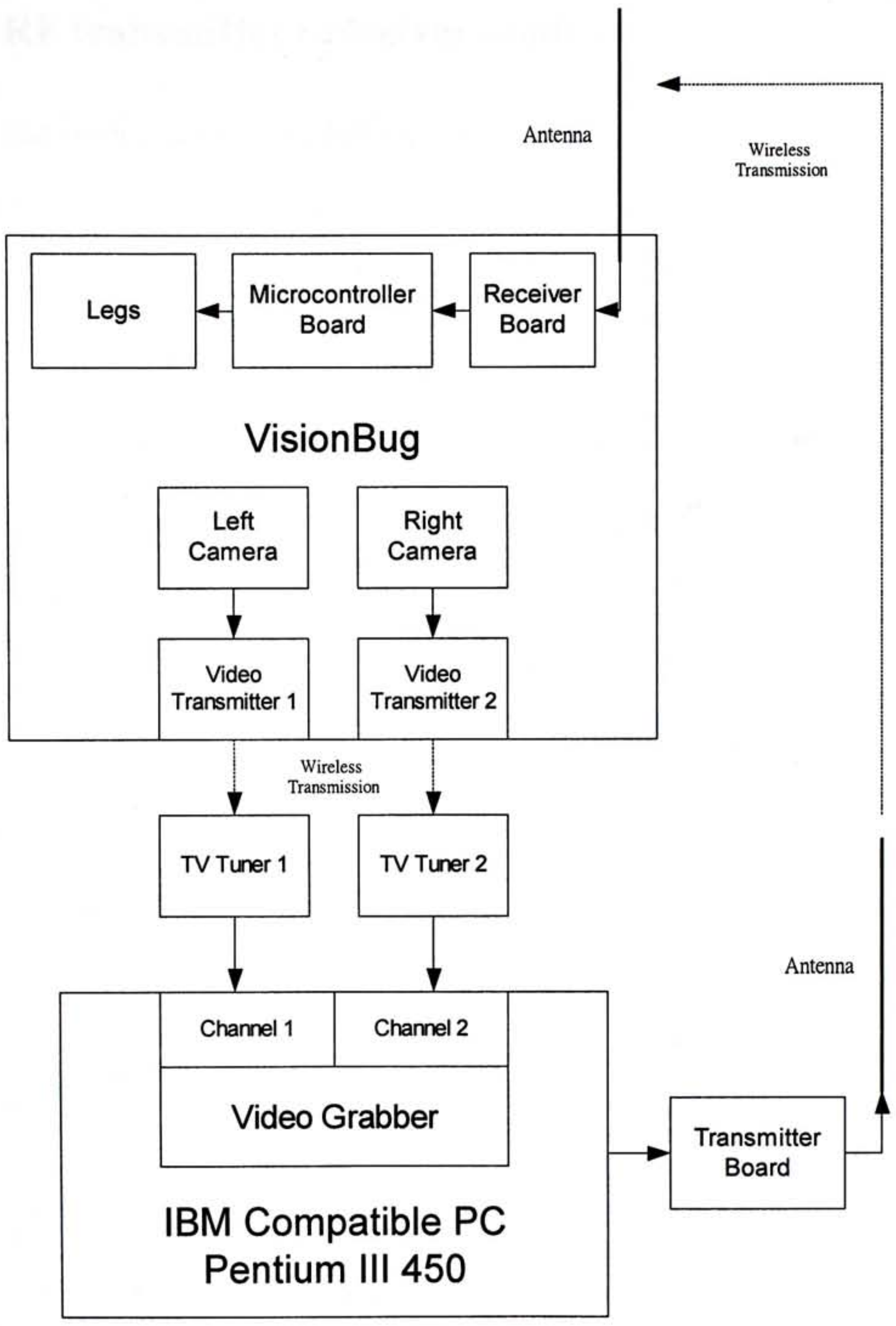


Fig.21 Hardware components and connections

6.1.2 RF transmitter / receiver modules:

The RF transmitter/receiver modules are part of our contribution, which are specially designed for commanding the robot wirelessly. In the market, there are different kinds of wireless transmission solution, such as wireless modem, wireless controller from R/C model cars. However, none of them are suitable to implement on our mobile platform. Most of the wireless modem in the market is relatively large compare to our small legged robot. Although there exists a wireless modem, which is small enough to fit into the robot, the modulation and demodulation processes will cause another problems. It is also not allowed in real-time robot control, since the delay time is not constant and increases the error of the predicted motion. So we used some high speed RF pulse transmission chips and make a direct RF transmission by means of interfacing to the RS232 signal.

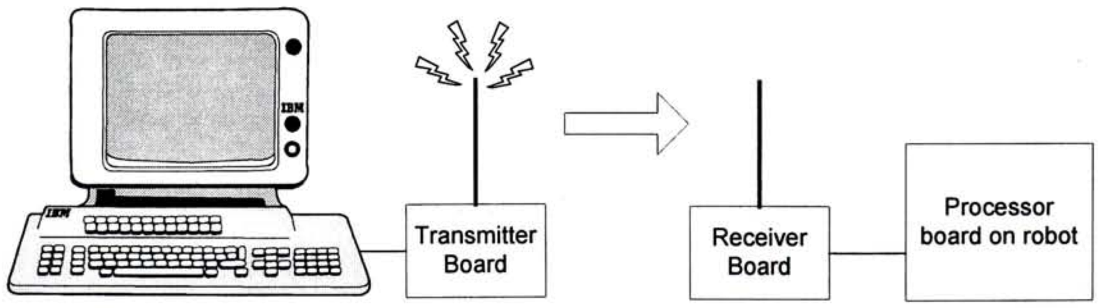


Fig.22 Connections of the RF transmission modules

Transmitter module:

The circuit diagram shows the transmitter circuit in 1:1 scale. The circuit inverts and converts the RS232 signal to CMOS level, and connects to the transmitter IC.

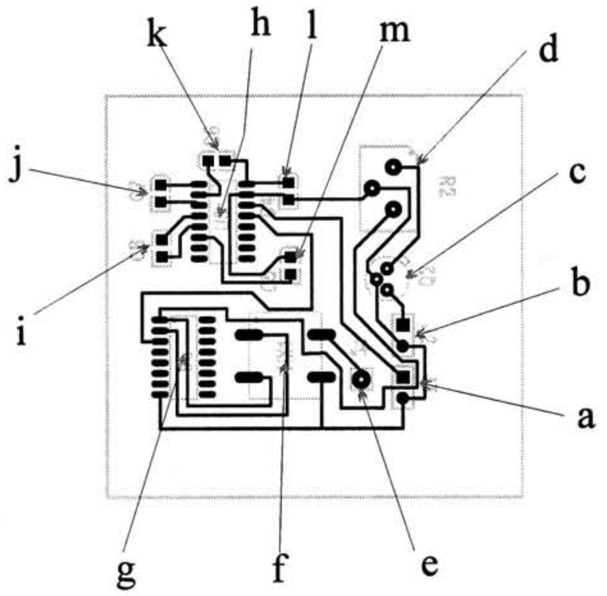


Fig.23 Serial data transmitter circuit board

Components in the circuit:

a	Power Voltage(Vcc)	+7 to +12V input.
b	Data Input	Connect to the serial port Tx and GND pins of the desktop computer.
c	LM7805 voltage regulator	Regulates the input voltage to +5V(Max. Current 100mA).
d	Trimmer	500 ohm trimmer.
e	Antenna	90mm long antenna.
f	HX2000	RF transmitter chip. Sample provided by RF Monolithics, Inc. (Data sheet see: http://www.rfm.com)
g	HCF4009UB	Hex CMOS level buffer/invertor
h	MAX232A	RS232 to TTF/CMOS signal convertor.
i	Capacitors	0.2 μF
j, k, l and m		0.1 μF

Table. 2 Transmitter board component list

Receiver module:

The circuit diagram below shows the receiver board in 1:1 scale. This receiver receives the signal transmitted from the transmitter. Since the data output is connected to the CMOS level serial port on the on-board processor, the received signal can be acquired by the on-board processor. Then, the one-way connection between the desktop computer and the mobile platform is established.

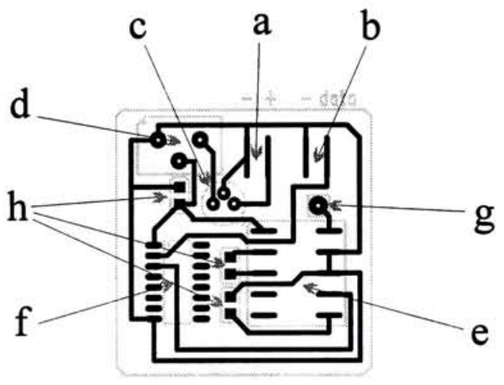


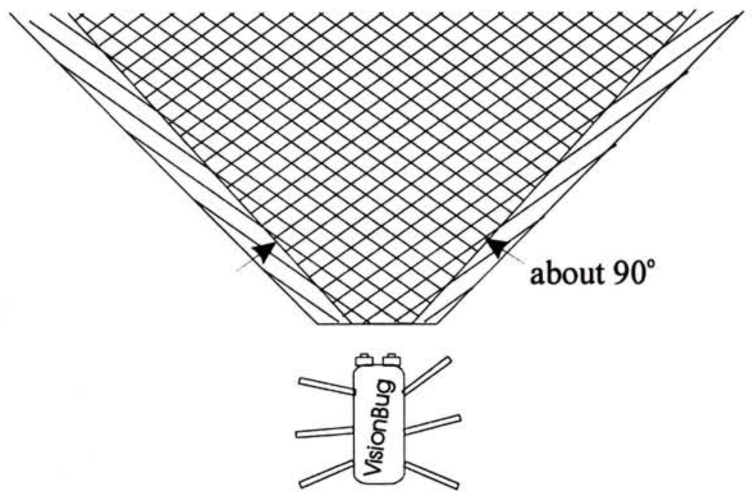
Fig.24 Serial data reciver board

a	Input voltage (Vcc)	+7 to +12 V
b	Data Input	Cmos level serial input +3.3V.
c	7805 voltage regulator	Regulate the input voltage to +5V (100mA)
d	Trimmer	Tune the input voltage to +3.3V
e	RX2056	RF receiver IC Sample provided by RF Monolithics, Inc. (Data sheet see: http://www.rfm.com)
f	HCF4009UB	CMOS level Hex invertor with buffer
g	Antenna	90 mm long antenna
h	Capacitors	10μF Surface mount capacitors

Table. 3 Receiver board component list

6.2 Perception

Two CMOS cameras with wide-angle lenses are attached to the front of the robot in order to cover larger field of view, and connected to two independent video transmitters. In our implementation, single type of sensor, the visual sensors (CMOS cameras) are mounted on the robot and have about 90° wide-angle stereo field of view. The view-able areas are shown in Fig.25. Since the cameras are in fixed resolution, the closer areas, corresponding to the lower part of the images, are having more detail than fara-way areas. The differences are more obvious if images are captured from wide-angle lens.



▨ Area covered by left camera

▧ Area covered by right camera

⊠ Area covered by both cameras

Fig.25 The viewable area of the VisionBug

6.3 Image Calibration

To increase the field of view of the robot, wide-angle cameras are mounted on the robot. The focal length of the lens is about 2.1mm, which is very wide-angle and close to a fisheye lens. Because of the effect of wide-angle lens, the captured images are highly distorted.

Fig.26 was an image captured from a square grid pattern using our camera and we can see the distortion in this image. Since we use homography based method, some intrinsic parameters of the camera can be combined into the homography. Also, there is no need to consider the extrinsic parameters the geometry of the two cameras with respect to the robot. Since the geometry is fixed. So, what we needed is only some intrinsic parameters, such as focal length, optical center and first and second order radial lens distortion coefficients. The full camera calibration is avoided.

According to Tsai's calibration method [45], (program code can be downloaded from http://www.cs.cmu.edu/afs/cs.cmu.edu/user/rgw/www/TsaiCode.html), we found the calibration data from the lens distortion.

In Fig.27, the regulated image is shown, and we can see that the patterns are close to the original grid pattern. However, the camera is so small that it is not easy to vertically face to the grid and the regulated image looks asymmetric. Also, since the resolution of the image sensor is limited, the center area will have better resolution than the area at the sides.

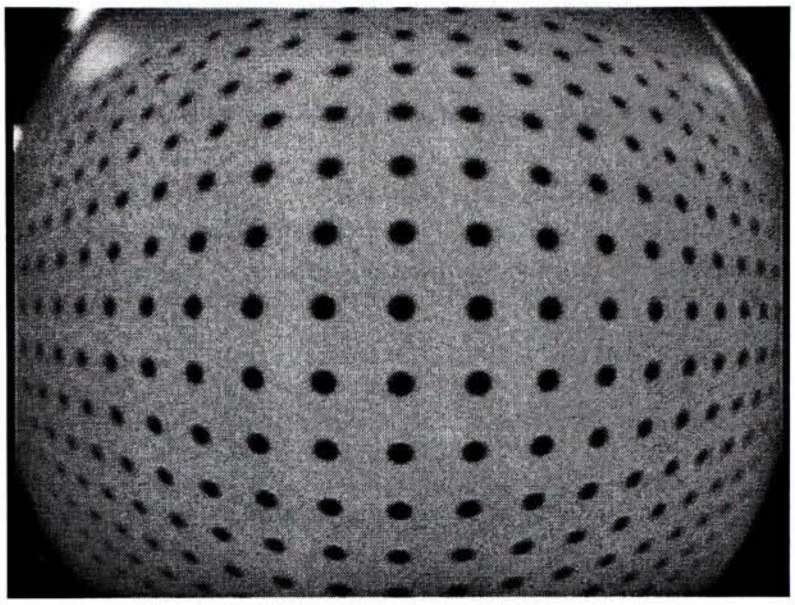


Fig.26 Image from square grid pattern

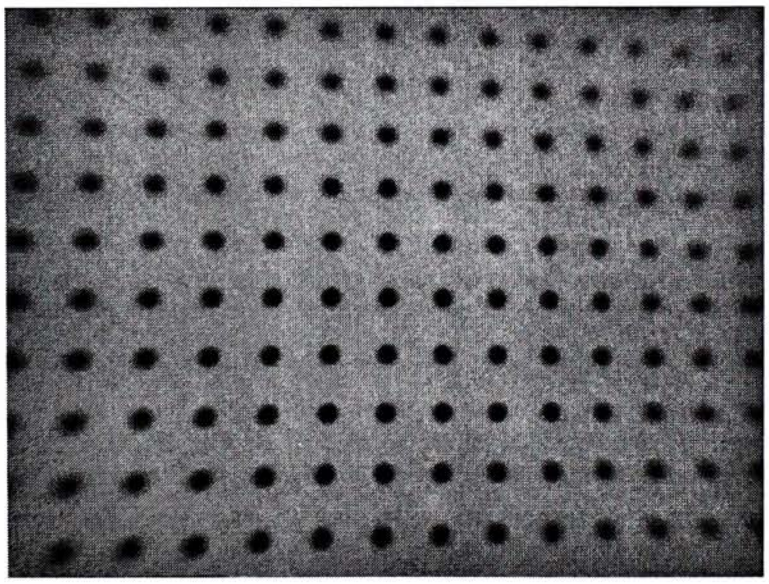


Fig.27 Regulated Image from calibration data

6.4 Motion Calibration:

Time delay is one of the problems in visually guided robot control. The controller has to control the robot from prediction of its environments. In visual control problems, the time delay will be much longer because of the computation intensive visual processes. By creating the local map and combine the transformation by the last command, we can predict the up-to-date local map in order to command the robot accurately.

Other than the map, we have to know how much the robot moved, the angle it rotated for any motion command. Therefore, motion calibration is necessary. Motion calibration is the step that recovers the mapping between the command and the real motion. Since the robot is legged and the motion is discrete, the calibration method is different with wheeled mobile robot. The method is described as below:

The model:

Assumes the motion of the robot is combination of rotation and translation on the ground plane. (See Fig.28) We defined two command values. One is the curvature command κ. It controls the ratio between the rotating speed and translation speed. Only it can change the trajectory of the robot motion. (The size of the circle it walking on.) The other command is the speed command ν. This command controls the speed of the robot. When the value is large, the robot moves faster. The route of the robot will not be changed by the speed command with the same κ.

These are the equations representing the model.

$$\begin{aligned} P(t+\Delta t) &= P(t) + R(\Omega(t))d(\nu,\kappa) \\ \Omega(t+\Delta t) &= \theta(\nu,\kappa) + \Omega(t) \end{aligned} \tag{9}$$

Notations:

$P(t)$	Position vector at time t
$\Omega(t)$	Orientation at time t in radian
t	Time
Δt	Time elapsed between two command sent (variant)
ν	Speed command
κ	Curvature command
θ	Angle rotated in single step
d	Displacement vector in single step

$$R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

In these equations, we attempted to find d and θ which depend on command ν and κ.

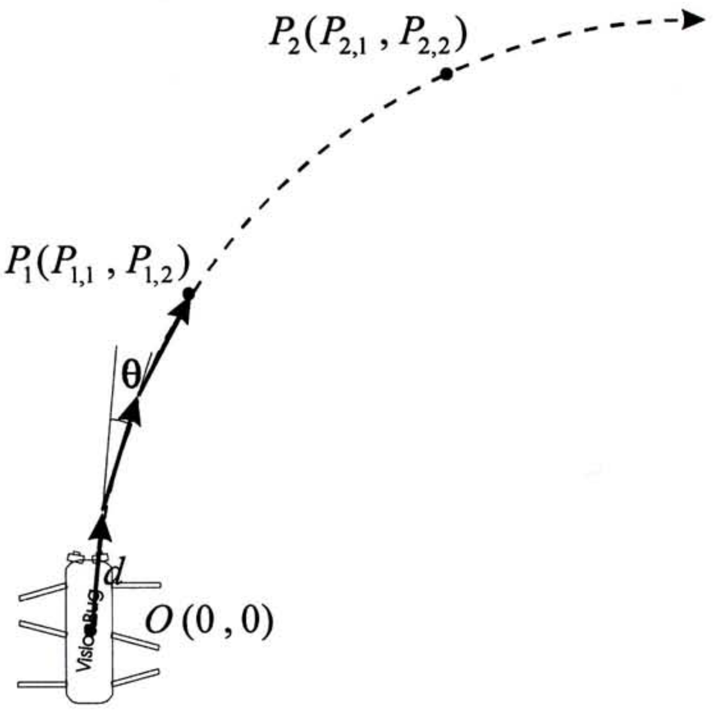


Fig.28 Path of robot mation with constant commands

Measurement of calibration data:

1. Define x and y-axes and position the robot to align with y-axis and the center of the cameras at the origin as Fig.28.
2. Command the robot to move $\boldsymbol{n}$ steps with the same v and κ commands. (where $\boldsymbol{n}$ is known and $\boldsymbol{n}$=5 in our experiments)
3. Measure the new position of the robot (distance moved in $\boldsymbol{x}$ and $\boldsymbol{y}$ directions). We call this P_1.
4. Repeat steps 2 & 3 and record P_2.
5. Repeat all the steps in different combination of the speed and rotation commands.

Calculations:

In this step, we are going to solve the d and θ with different commands. Let P_1 and P_2 are the measured position vectors. Base on the model mentioned above. We have:

$$P_1 = \sum_{i=0}^{n-1} R(\theta)^i d \qquad\qquad P_2 = \sum_{i=0}^{2n-1} R(\theta)^i d \tag{10}$$

Such that:

$$P_2 - P_1 = R(\theta)^n \sum_{i=0}^{n-1} R(\theta)^i d = R(\theta)^n P_1 \tag{11}$$

Then, from the angle between vectors P_1 and $P_2 - P_1$:

$$\theta(\nu,\kappa) = \frac{1}{n}\operatorname{acos}\left(\frac{P_1^T(P_2 - P_1)}{\|P_1\|\cdot\|P_2 - P_1\|}\right) \tag{12}$$

$$d(\nu,\kappa) = \left(\sum_{i=0}^{2n-1} R(\theta)^i\right)^{-1} P_2 = \left(I - R(\theta)\right)\left(I - R(\theta)^{2n}\right)^{-1} P_2 \tag{13}$$

From the equations above, we have measured one data set for each variable. They are plotted in the graphs below. The first one is the $\theta(\nu,\kappa)$. We can easily see that it is close to a straight line. We can easily approximate their relationship by linear regression.

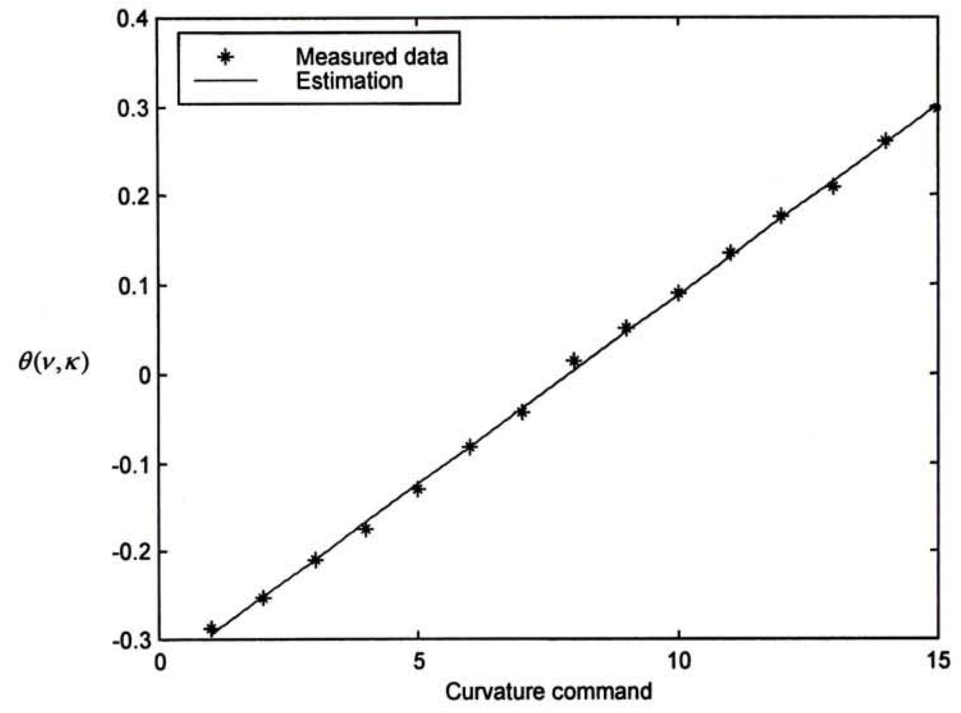


Fig.29 Calibration: curvature command vs angle rotated

By linear regression, the data is approximated by the line:

$$\theta(\kappa,\nu) = (-0.34043081515815 + 0.04257533090013\kappa)\nu$$

Max error = 0.01138330201773 radian (0.65221516253848 degree)

The next one is to find the x component of the vector *d*. The graph below shows the plot of the calibration data and a fitting line. Since these values are small, it makes the error looks so large. Indeed, the maximum error is about 5 mm.

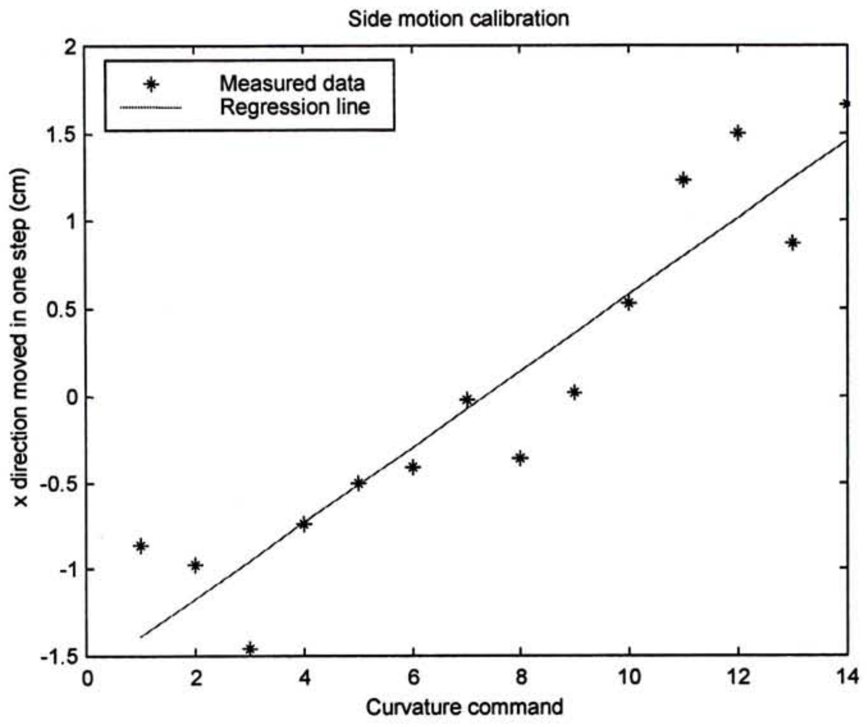


Fig.30 Calibration: Curvature command vs x distance moved

By linear regression, we found:

$$d_x = (-1.60828209014281 + 0.21895534641356\kappa)\nu$$

Max error = 0.528cm

The last parameter is the y component of d. We can found that it can be fit into two straight lines. Also by regression, two lines are found.

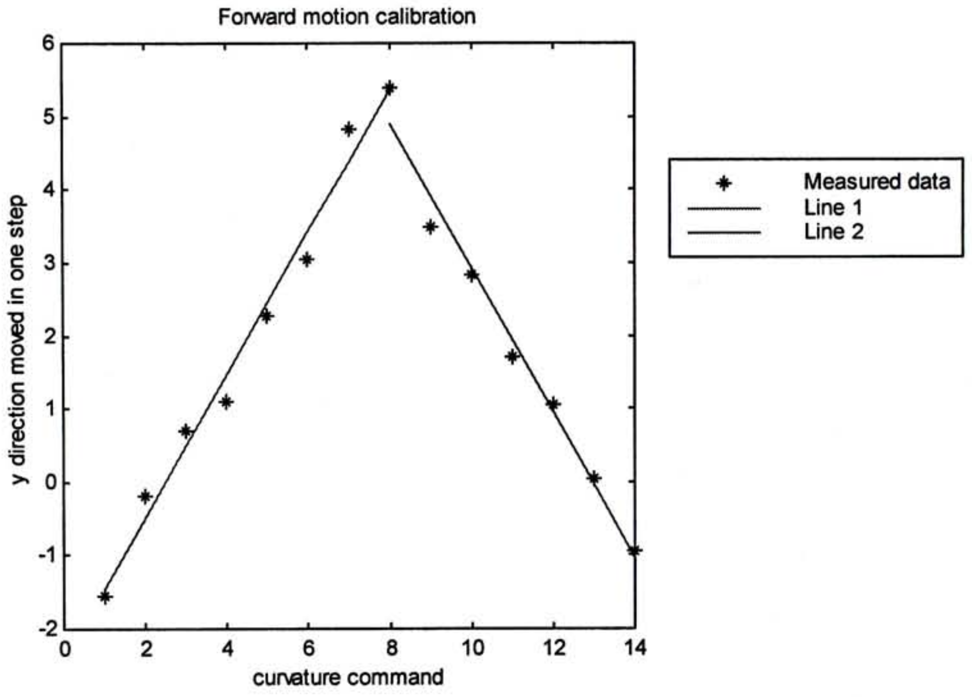


Fig.31 Calibration: Curvature command vs y distance moved

By divide the data and apply linear regression, we found two lines:

$$d_y = (-2.44325174527432 + 0.97588853433808\kappa)v \qquad (\kappa \le 8)$$
$$d_y = (12.81351572244978 - 0.98858945100810\kappa)v \qquad (\kappa > 8)$$

Max error = 0.44419cm

After calibration, real motion can be found by sending the same command in ***n*** cycles:

$$P(n) = \sum_{i=i}^{n-1} R(\theta)d = \left(I - R(\theta)\right)^{-1}\left(I - R(n\theta)\right)d$$

Replace ***n*** by ***t/T***, we have motion in time (***t***):

Let $c_1 = \cos\theta$, $s_1 = \sin\theta$, $c_2 = \cos\dfrac{t\theta}{T}$ and $s_2 = \sin\dfrac{t\theta}{T}$:

$$P(t) = \left(I - R(\theta)\right)^{-1}\left(I - R(\frac{t}{T}\theta)\right)d$$

$$= \begin{bmatrix} 1-c_1 & -s_1 \\ s_1 & 1-c_1 \end{bmatrix}^{-1} \begin{bmatrix} 1-c_2 & -s_2 \\ s_2 & 1-c_2 \end{bmatrix} \begin{bmatrix} d_x \\ d_y \end{bmatrix}$$

$$= \frac{1}{2-2c_1} \begin{bmatrix} 1-c_1 & s_1 \\ -s_1 & 1-c_1 \end{bmatrix} \begin{bmatrix} 1-c_2 & -s_2 \\ s_2 & 1-c_2 \end{bmatrix} \begin{bmatrix} d_x \\ d_y \end{bmatrix}$$

$$= \begin{bmatrix} \left[(1-c_1)(1-c_2) + s_1 s_2\right]d_1 + \left[s_1(1-c_2) - s_2(1-c_1)\right]d_x \\ \left[s_2(1-c_1) - s_1(1-c_2)\right]d_1 + \left[(1-c_1)(1-c_2) + s_1 s_2\right]d_y \end{bmatrix} \qquad (14)$$

6.5 Software Programs

According the designed architecture of the system, two programs are developed. One of them is running on the robot and the other is running on the PC with frame grabber. The program running on the PC processes all the vision calculations. The fuzzy controller is also implemented in this program. This program processes the input images as described in Chapter 4 and implements the fuzzy controller as in Chapter 5. In Fig.32, the detail processes in the two programs are shown. On the right of the figure, we can see the program, which run on the PC. The processes in this program include feature extraction, image regulation, recursive homography estimation and the fuzzy controller. The programming language used is Microsoft Visual C++ 6.0 in most of the calculations. Also, this program used Matrox MIL Image Processing Library 5.0 in capturing image and handle some kernel filtering.

On the left of the figure, it shows the program running on the micro-controller, which controls the robot. This program's main usage is to communicate with the remote PC, get the command and assign the gait pattern to the legs. The programming language used in this program is the GNU C++.

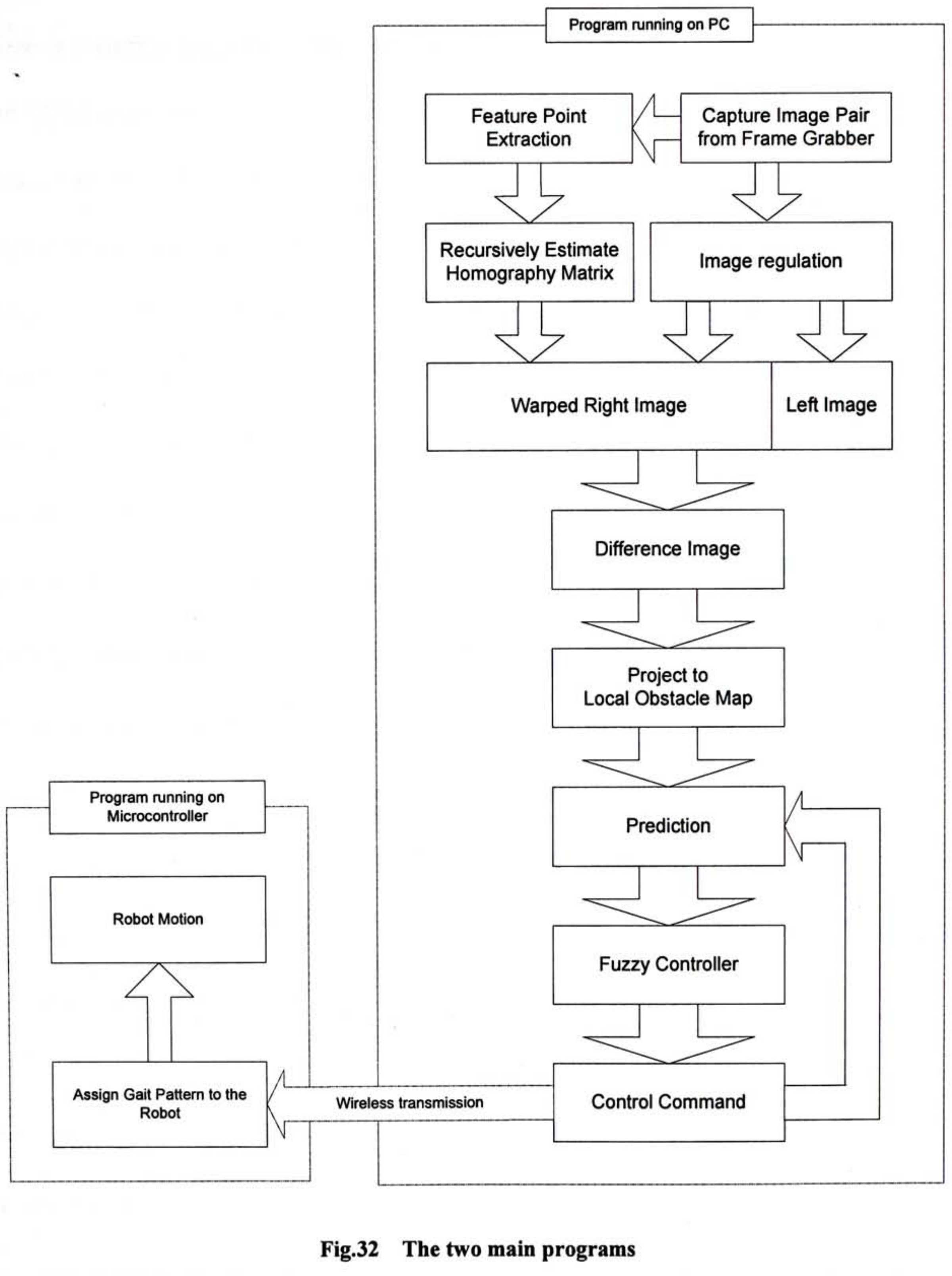


Fig.32 The two main programs

6.5.1 Computational complexity

Here is a simple analysis of the computational complexity of the processes that update the LOM from time $(t - \Delta t)$ to time t. The computations for extracting features from images are excluded, since it is not the contribution of this work, any feature extraction algorithms are all applicable in this work.

Suppose P features are detected in each image and F point correspondences are matched in a pair of stereo images. In searching for point correspondences, since we already know the previous ground homography $H_{t-\Delta t}$ at time $t - \Delta t$, and we assume that the new homography at time t is close to it, we only search along the epipolar line in a small neighborhood of the expected position estimated from $H_{t-\Delta t}$. The entire correspondence establishment process therefore costs *O(P)* time. After the extraction of the correspondences, the matrix $\hat{M}$ in equation (6) is formed and the eigenvalues and eigenvectors of the 9 by 9 matrix $\hat{M}^T\hat{M}$ are calculated. Including input point normalization, finding the solution of H recursively takes up $O(P) + O(P^2) = O(P^2)$ time, as the size of the matrix $\hat{M}^T\hat{M}$ is fixed.

In the construction of the LOM, assuming the image and the map resolution are $N \times N$ and $R \times R$ respectively, the warping of the right image to the left view and the comparison made there take a total of $O(N^2)$ time. Finally, the warping of the obstacle image to LOM is a process of $O(R^2)$.

In total, it takes $O(P^2) + O(N^2) + O(R^2)$ time to find the obstacle map from detected features.

Chapter 7

Experimental Results

We have tested our robot with a textured environment where the paths, the walls, the small hills, and dead-ends, which with the same textures, can be arranged as wished. Fig.33 and Fig.34 show some navigation steps of the robot made in every 12 seconds. The snapshots are read from left to right, top to bottom.

The robot successfully walked between two walls without crashing to them, and made a turn-around at a dead-end. Empirical data has shown that the legged motion does cause a large disturbance to the geometry between the robot and the ground surface, and in turn to the homography between the stereo images. An assumption of a constant plane-transformation between the images, which is widely proposed for wheeled robots, would certainly cause failure in capturing the obstacles. We believe the success of our system is due mostly to the allowance that the homography between the images can change over time.

7.1 Real Navigation Experiments

In the experiments of the real navigation, we can see that the robot moves very slowly. The control command is sent every 2 to 3 seconds. It is mainly caused by the slow computation of an IBM PC. There still also some reasons affecting the global performance:

- Wide angle camera lenses:

  In real experiment, we used wide-angle camera lenses in order to increase the field of view of the obstacle detection. However, other than more difficult camera calibration, wide-angle lenses caused other problem that was not predicted.

  When we apply the method of Homography, model of pinhole camera must be employed. However, the stereo images we captured from usual cameras and lenses, must be previously rectified in order to simulate images captured from pinhole cameras. After regulation, it is obvious that the areas far from the image center are blurred, especially at the corners. This kind of distortion is very usual and obvious after regulation in wide-angle lenses. Since some area is blurred in the image, this also means the effective resolution as well as the textureness are also affected. Then the image coordinate of detected feature on the corners contains a high error. It directly affects the accuracy of the calculated homography.

- Video RF transmission

  Another factor affecting the experiment is bad video transmission. Since the RF video transmitters use reflective wave in the range of UHF and VHF, which are mainly used in TV broadcasting. This kind of radio wave highly depends on the geometry of the antenna, so that every TV antenna needed to be placed for the best picture quality.

The bad video transmissions result on the captured images. The affected image will be distorted or ruined. If this effect remains for a few iterations, the robot would probably be like a blind insect and crashes into obstacles. Fortunately, this bad video transmission would not last long in real experiments. If it just happens in a short period, the system can overwhelm this effect by the probabilistic LOM.

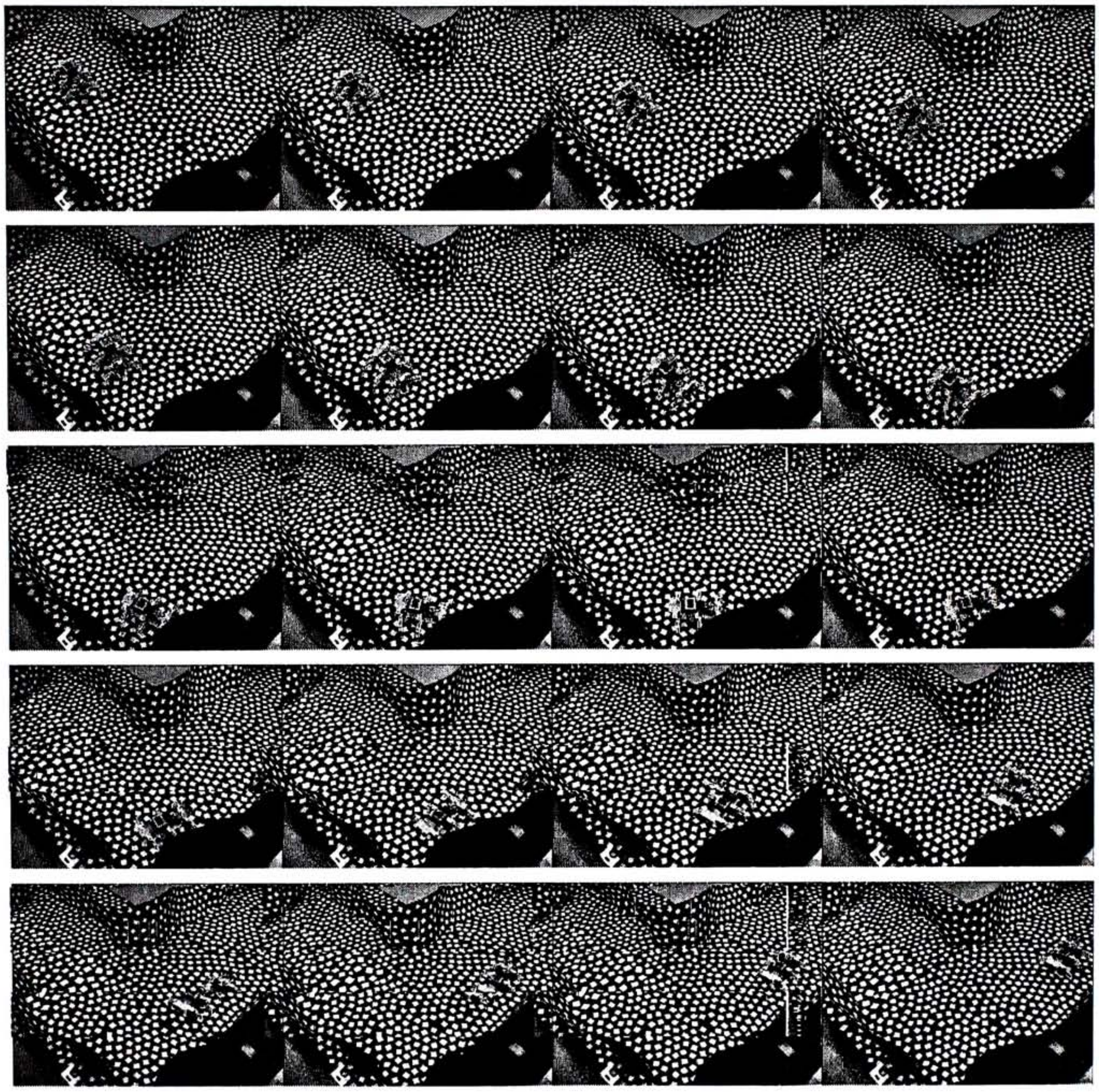


Fig.33 Experiment results – Turning in a left corner

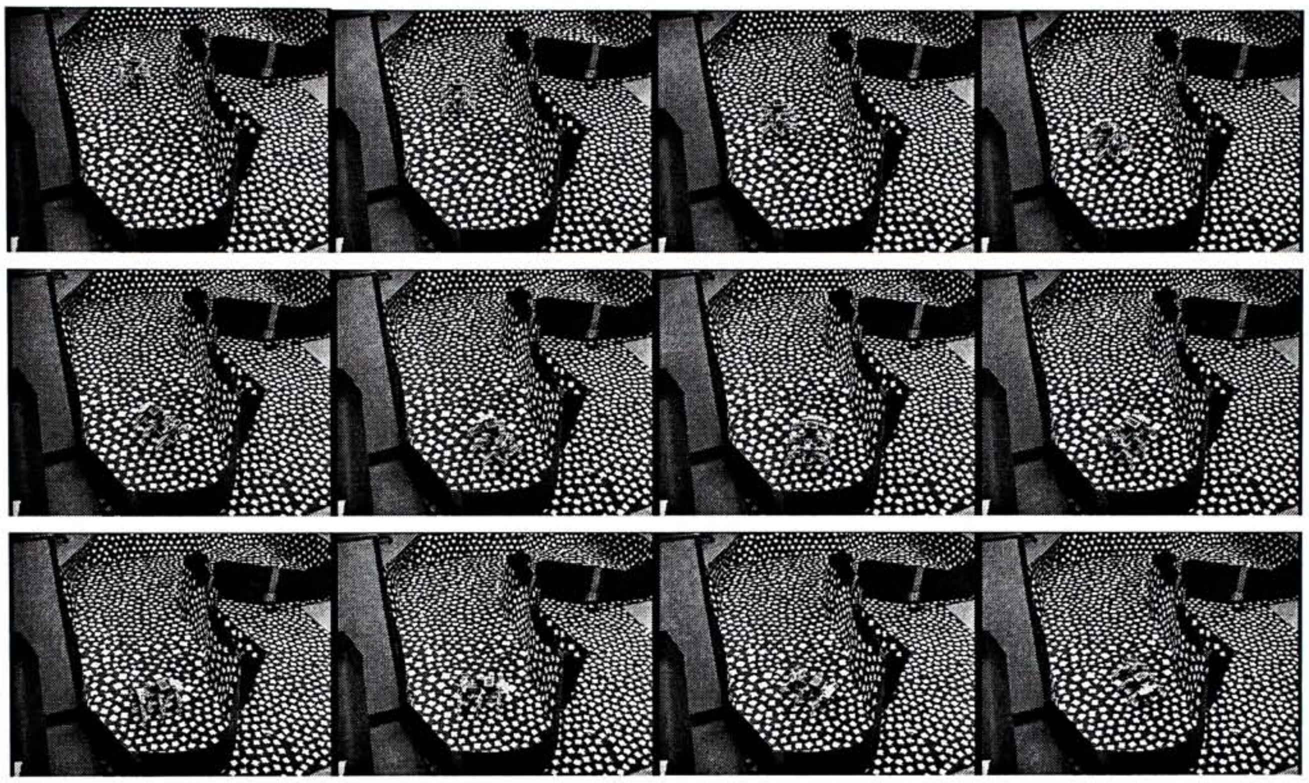


Fig.34 Experiment results – Withdrawing from dead-end

7.2 Error Analysis of LOM

In the last sub-section, experiment shows that the robot can navigate inside a closed environment. However, no quantitative evaluation is taken in order to analyze the accuracy of the LOM. Here, we will conduct a simple test to analyze the error of the detected obstacle distance. The procedures are shown as follows: First, we put a textured cardboard at the front of the robot. And we measure a fixed and known distance from the obstacle. Then, we program the robot to move to the front. Until the robot arrived to the fixed distance, we capture the obstacle image. Then, according to the obstacle detected in the image, we can reproject the position of the obstacle from the image coordinates to the ground coordinates. And we can also know the detected obstacle distance. Tables 4. to 7. below show 4 sets of data taken at 4 different obstacle distance. Each table contains detected obstacle distance at each fixed obstacle distance:

Detected Obstacles' *y* coordinates (pixels)	Calculated Obstacle Distance (cm)	Error (cm)
219	12.7342	2.7342
227	11.5650	1.5650
229	11.2901	1.2901
233	10.7595	0.7595
228	11.4268	1.4268
234	10.6306	0.6306
228	11.4268	1.4268
230	11.1551	1.1551
	Mean Square Error	1.4987

Table. 4 A wall positioned at 10cm in front of the robot

Detected Obstacles' *y* coordinates (pixels)	Calculated Obstacle Distance (cm)	Error (cm)
179	20.9551	0.9551
184	19.6354	-0.3646
174	22.3914	2.3914
172	23.0021	3.0021
169	23.9605	3.9605
174	22.3914	2.3914
180	20.6823	0.6823
173	22.6940	2.6940
	Mean Square Error	2.3692

Table. 5 A wall positioned at 20cm in front of the robot

Detected Obstacles' *y* coordinates (pixels)	Calculated Obstacle Distance (cm)	Error (cm)
126	47.3979	7.3979
130	43.9880	3.9880
138	38.2290	-1.7710
127	46.5068	6.5068
134	40.9510	0.9510
132	42.4268	2.4268
138	38.2290	-1.7710
136	39.5538	-0.4462
	Mean Square Error	3.9724

Table. 6 A wall positioned at 40cm in front of the robot

Detected Obstacles' *y* coordinates (pixels)	Calculated Obstacle Distance (cm)	Error (cm)
103	79.5660	-0.4340
94	104.3060	24.3060
97	94.6904	14.6904
100	86.5487	6.5487
99	89.1203	9.1203
93	107.9101	27.9101
89	124.8872	44.8872
105	75.4381	-4.5619
	Mean Square Error	21.6428

Table. 7 A wall positioned at 80cm in front of the robot

From the result, we can see that the error increases exponentially as the obstacle distance increases. This increasing error is explained by the significance of the pixel of the image captured. As each image point represents a fixed angle of perception, the image points at the bottom of the image cover lesser area of the ground. The image resolution to the ground is higher at the closer portions. Like any other animals' vision, far away objects are not so clear as close objects. This also is the fact that all types of sensors could not avoid.

Chapter 8

Conclusion and future work

In this thesis, a visually guided obstacle detection and avoidance system on a legged robot platform has been presented. The system detects obstacles, such as walls or small hills or dead-ends, on a roughly textured planar surface. The obstacle information obtained is sufficient to drive the robot to move about using a simple fuzzy controller. Hardware implementation has been conducted and experiments have been performed which show that the robot can walk in real-time without crashing to the obstacles.

For the part of obstacle detection, an image-to-image mapping called homography induced by the ground across the two images is first estimated. The homography allows the right image to be warped to the left view. The original left image and the warped right image would be consistent over features on the ground, but not over features not on the ground. Information about the obstacles can thus be extracted on the left view, and such a 2D obstacle distribution can be warped to the ground through another homography, which in turn allows obstacle information on the ground be extracted in terms of a 2D map we refer to as the Local Obstacle Map (LOM). The essence of the approach is that an explicit 3D reconstruction of the surroundings is no longer necessary.

A simple and classical fuzzy controller can then be used to command the robot's global motion, the speed and steering, based on the LOM information. By implementing the suitable rules to the rule table, the robot can avoid large obstacle, such as walls, and small hills. Based on different kind of environment, it will behave to turn left, turn right and reverse.

Although the designed fuzzy controllers are easy to implement but it still cannot optimally control the robot. To obtain the optimal performance, the controller should have the adaptive ability or unsupervised learning ability.

References

[1] N. Ayache and F. Lustman, Trinocular stereo vision for robotics, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.13, no.1, Jan 1991, pp73-85.

[2] C. Barret, M. Benreguieg and H. Maaref, Fuzzy agents for reactive navigation of a mobile robot, Proceedings of Int. Conf. on Knowledge Based Intelligent System, Adélaide, 21-23 May 1997, vol.2, pp.649-658.

[3] Randall D. Beer, Roger D. Quinn, Hillel J, Chiel, and Roy E. Ritzmann. Biologically Inspired Approaches to Robotics. Communications of the ACM, March 1997 Vol. 40 No.3 pp.31-38.

[4] A. Benedetti and P. Perona, Real-time 2-D Feature Detection on a Reconfigurable Computer. Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1998, pp.586-593.

[5] M. Benreguieg, P. Hoppenot and H. Maaref, Fuzzy navigation strategy: application to two distinct autonomous mobile robots, Robotica, 1997, vol.15, pp.609-615.

[6] M. Bertozzi, A. Broggi and A. Fascioli, Real-Time Obstacle Detection using Stereo Vision, In Proceedings EUSIPCO-96 - VIII European Signal Processing Conference, Trieste, Italy, September 10-13 1996.

[7] A. Blake, J.M. Brady, R. Cipolla, Z. Xie and A. Zisserman, Visual navigation around curved objects, Proceeding of IEEE International Conference on Robotics and Automation, 1991, pp.2490-2495.

[8] Wolfram Burgard, Dieter Fox, Daniel Hennig and Timo Schmidt, Estimating the Absolute Position of a Mobile Robot Using Position Probability Grids, Proc. Of the 13th National Conference on Artificial Intelligence, August 4-8, 1996, Portland, Oregon.

[9] S. Carlsson and J.O. Eklundh, Object Detection Using Model Based Prediction and Motion Parallax. Preceedings of ECCV'90, LNCS, pp.134-38, 1990.

[10] Bing-Yung Chee, Sherman Y.T. Lang and Peter W.T. Tse, Fuzzy Mobile Robot Navigation and Sensor Integration, Fuzzy Systems, 1996., Proceedings of the Fifth IEEE International Conference on , Volume: 1 , 1996 , pp. 7-12.

[11] Y. H. Chow, R. Chung, Obstacle Avoidance of Legged Robot without 3D Reconstruction of the Surroundings, Technical Report, Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, 1999.

[12] R. Chung and R. Nevatia, Recovering Building Structures from Stereo, Proceedings of the IEEE Workshop on Applications of Computer Vision, pp. 64-73, Palm Spring, CA. USA, 1992.

[13] S.D. Cochranand, G. Medioni, 3-D Surface Description from Binocular Stereo, IEEE Transactions on Pattern Recognition and Machine Intelligence, 14(10):981-994, Oct. 1992

[14] M. Eibert, P. Lux and C. H. Schaefer, Autonomous obstacle avoidance for off-road vehicles, Proceeding of 2nd International Conference on Intelligent and Autonomous Systems, Amsterdam, Netherlands, 1990, pp.737-747.

[15] W. Enkelmann, Obstacle Detection by Evaluation of Optical Flow Fields From Image Sequences. Image and Vision Computing, vol.9, no.3, pp.160-8, 1991.

[16] O. Faugeras, Stratification of three-dimensional vision: projective, affine, and metric representations, Journal of the Optical Society of America - A, vol. 12, pp.465-484, Mar. 1995.

[17] O. Faugeras, What Can be Seen in Three Dimensions with an Uncalibrated Stereo Rig?, Proceedings of European Conference on Computer Vision, pp. 563-578, Santa Margherita Ligure, Italy, May 1992.

[18] F. Ferrari, E. Grosso, G. Sandini and M. Magrassi, A stereo vision system for real time obstacle avoidance in unknown environment, Proceedings. IROS '90. IEEE International Workshop on Intelligent Robots and Systems '90, vol.2, 1990, pp.703-8.

[19] P. Fornland, Direct Obstacle Detection and Motion from Spatio-Temporal Derivatives, Proceedings of CAIP'95, LNCS, pp874-9, Prague, September 1995.

[20] J. Gaspar, J. Santos-Victor, J. Sentieiro, Ground Plane Obstacle Detection with a Stereo Vision System, International Workshop on Intelligent Robotic Systems - IRS94, Grenoble, France, July 1994.

[21] J.J. Gibson, The Perception of the Visual World, Houghton-Mifflin, Boston, 1950.

[22] Richard I. Harley; In Defense of the Eight-Point Algorithm, IEEE Transactions on Pattern Analysis and machine Intelligence, vol.19, no.6, June 1997, pp.580-593.

[23] H. Hu, Dynamic Planning and Real-time Control for a Mobil Robot, Ph.D. thesis, University of Oxford, 1992.

[24] H. Ishiguro, K. Kato, S. Tsuji, Multiple vision agents navigating a mobile robot in a real world, Proceedings IEEE International Conference on Robotics and Automation, IEEE Comput. Soc. Press, Los Alamitos, CA, USA., vol.1, 1993, pp.772-777.

[25] M.R.M. Jenkin, and A. Jepson, Detecting Floor Anomalies, Preceedings of BMVC'94, pp.731-40, 1994.

[26] G. A. Jones; Constraint, Optimization, and Hierarchy: Reviewing Stereoscopic Correspondence of Complex Features, Computer Vision and Image Understanding, Jan. 1997, vol.65, no.1, pp.57-78.

[27] W. Krueger, W. Enkelmann and S. Roessle, Real-time estimation and tracking of optical flow vectors for obstacle detection, Proceedings of the IEEE Intelligent Vehicles '95. Symposium, New York, USA, 1995, pp.304-309.

[28] J.J. Leonard and H.F. Durrant-Whyte, Application of multi-target tracking to sonar-based mobile robot navigation, Proceeding of the 29th IEEE Conference on Decision and Control, Hawaii, 1990, pp.3118-3123.

[29] Fuxing Li and Michael Brady, Modeling the Ground Plane Transformation for Real-Time Obstacle Detection, Computer Vision and Image Understanding, vol.71, no.1, July, pp.137-152, 1998.

[30] S. H. Lian, Fuzzy Logic Control of an Obstacle Avoidance Robot, Proceedings of the Fifth IEEE International Conference on Fuzzy Systems, 1996, vol.1, pp.26 –30.

[31] Manolis I. A. Lourakis and Stelios C. Orphanoudakis; Visual Detection of Obstacles Assuming a Locally Planar Ground; Lecture Notes in Computer Science 1352, Third Asian Conference on Computer Vision, Hong Kong, China, January 1998 Preceedings, vol.2, pp.527-34, 1998.

[32] S.Y. Lu and Y.S. Fu, A Sentence-to-Sentence Clustering Procedure for Pattern Analysis, IEEE Transactions on Systems, Man and Cybernetics, 8(5):381-389, May 1978.

[33] Q.T. Luong, and T. Vieville, Canonic Representation for the Geometries of Multiple Projective Views, Computer Vision and Image Understanding, 64(2):194-229, Sept. 1996.

[34] M. Maruyama and S. Abe, Acquiring a Polyhedral Structure Through Face Extraction and Verification, Proceedings of International Conference on Pattern Recognition, pp. 579-581, Rome, Italy, Nov. 1988.

[35] I. Masaki, Vision-Based Vehicle Guidance, Springer-Verlag, Berlin, 1992.

[36] R. Mohan and R. Nevatia, Using Perceptual Organization to Extract 3D Structures, IEEE Transactions on Pattern analysis and Machine Intelli- gence, 11(11):1121-1139, Nov. 1989.

[37] P.J. Rousseeuw, Least Median of Squares Regression, Journal of American Statistics Association, vol.79, pp.871-80, 1984.

[38] M. Rygol, P. McLauchlan, P. Courtney, S. Pollard, J. Porrill and C. Brown, Parallel 3D Vision for Vehicle Navigation, IOS Press, 1991.

[39] A. Saffiotti, K. Konolige and E.H. Ruspini, A multivalued Logic Approach to Integrating Planning and Control, Artificial Intelligence, vol.76, 1995.

[40] J. Santos-Victor and G. Sandini; Uncalibrated Obstacle Detection Using Normal Flow, Machine Vision and Applications, vol.9, no.3, pp.130-7, 1996.

[41] A. Shashua, Projective Structure from Uncalibrated Images: Structure from Motion and Recognition, IEEE Transactions on Pattern Analysis and Machine Intelligence, 16(8):778-790, Aug. 1994.

[42] A. Shashua, Algebraic Functions for Recognition, IEEE Transactions on Pattern Analysis and Machine Intelligence, 17(8):779:789, August 1995.

[43] K-T Song and L-H Sheen, Fuzzy-neuro Control Design for Obstacle Avoidance of a Mobile Robot, 1995. International Joint Conference of, the Fourth IEEE International Conference on Fuzzy Systems and The Second International Fuzzy Engineering Symposium., Proceedings of 1995, IEEE International Conference on Fuzzy Systems, Volume: 1 , 1995 , pp. 71-76.

[44] C. Tomasi and T. Kanade, *Detection and Tracking of Point Features*, Tech. Rep. CMU-CS-91-132, Carnegie Mellon University, Apr. 1991.

[45] R. Y. Tsai, *An efficient and accurate camera calibration technique for 3D machine vision*, in IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 364–374, 1986.

[46] V. Venkateswar and R. Chellappa, Hierarchical Stereo Matching Using Feature Groupings, Proceedings of the DARPA Image Understanding Workshop, pages 427-436, San Diego,California, Jan. 1992.

[47] T. Williamson and C. Thorpe, A Specialized Multibaseline Stereo Technique for Obstacle Detection, Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, June, 1998, pp.238.

[48] M. Xie, Ground plane obstacle detection from stereo pair of images without matching, Proceedings of ACCV '95. Second Asian Conference on Computer Vision, Singapore, vol.2, 1995, pp.280-284.

[49] G.S. Young, M. Herman, T.H. Hong, D. Jiang and J. Yang; New Visual Invariants for Terrain Navigation Without 3D Reconstruction, International Journal of Computer Vision, 1998, vol.28, no.1, pp.45-71.

[50] M. Zhang, S. Peng and Q. Meng, Neural network and fuzzy logic techniques based collision avoidance for a mobile robot, Robotica, vol.15, 1997, pp.627-632.

[51] Z. Zhang, Determining the Epipolar Geometry and Its Uncertainty: A Review, International Journal of Computer Vision, Mar. 1998, vol.27, no.2, pp.43-76.

[52] Z. Zhang, R. Weiss and A.R. Hanson, Obstacle Detection Based on Qualitative and Quantitative 3D Reconstruction, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.19, no.1, pp.15-26, Jan. 1997.

[53] Y. Zheng, D.G. Jones, S.A. Billings, J.E.W. Mayhew and J.P. Frisby, A stereo algorithm for ground plane obstacle detection, Image Vision Computing, vol.8, no.1, Feb 1990, pp.57-62.