Vision and path planning in RoboCup

Wang, Chunmiao

2006

Wang, C. (2006). Vision and path planning in RoboCup. Master's thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/3645>

<https://doi.org/10.32657/10356/3645>

Nanyang Technological University

Downloaded on 26 Apr 2023 18:39:50 SGT

Vision and Path Planning in RoboCup

Wang Chunmiao

School of Electrical & Electronic Engineering

A thesis submitted to the Nanyang Technological University
in fulfillment of the requirement for the degree of
Master of Engineering

2003

T
3
WCM
2003

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research and has not been submitted for a higher degree to any other University or Institution.

10/04/2023

Date

Wang Chunmiao

Wang Chunmiao

Acknowledgements

I am very grateful to the School of Electrical and Electronic Engineering of Nanyang Technological University for offering the scholarship and the research facilities to complete my thesis. I would like to thank my thesis supervisor, Professor *Soh Yeng Chai* and co-supervisor, Professor *Wang Han*, for their continual support and encouragement throughout my candidature. I really appreciate their assistance.

Special thanks also go to our laboratory technicians, Mr. *Chia Chiu Kiat* and Mr. *Tan Chai Soon* for their relentless effort in logistic and administrative supports. I would like to thank Mr. *Wang Hui* for his cooperation in this RoboSoccer Project. I am grateful for the assistance from *Zou Yi*, *Zhou Xiaowei*, *Yu Tieniu*, *Zhou Weiqing*, *Sarath*, and *Chen Cheng* in conducting my experiments. We study together, play together, and even eat together. I have almost spent two carefree and happy years of my life with them.

Finally, I would like to express my deepest gratitude to my parents and my wife, who are always so supportive in my works and give me the strength and motivation to move forward in my life.

Summary

The Robot World Cup Initiative (RoboCup) is an international joint project to promote Computer Vision, AI, Robotics, Motion Planning, and related fields. It provides a standard platform for a robotic soccer game. It is a combination of engineering application and theoretical development. It is a great honor to join this project and work on parts of the work. This thesis can be divided into two parts:

1. First part is about the design and implementation of the Vision Module in RoboCup.
2. The second part is about path planning with obstacle avoidance.

The vision system can be divided into three parts: (a) real-time tracking, (b) camera calibration; (c) color calibration. The aim of the vision system for the RoboCup small robot league is to track and predict the motion states of 10 agents and a ball, and send the states information to the Artificial Intelligence module. This thesis presents the design and implementation of a real-time, robust global vision system. The work on the tracking system includes: a color-based segmentation for locating the interesting objects in the image; a blob analysis for calculating the positions of the objects in the image and filtering out the noise; a double-check data association for associating the current time frame information with the previous time frame information; the identification of role for calculating the orientation and correct some errors of data association, and a linear filter for tracking and predicting the states information; an UDP socket for transmitting the information from the vision system to the AI system. This thesis also presents the K-

Mean-clustering-based color calibration and Tsai's camera calibration model. Experimental results show the effectiveness of the proposed scheme.

The path planning in this thesis does not deal with practical problems of a real competition, where the environment is dynamic, but it is of theoretical interest to study the nature of a simplified static path planning problem. A near optimal path planning algorithm is presented in a static environment. In the thesis, a new hierarchical genetic algorithm (**HGA**) for path planning in a static environment with obstacles is presented. The algorithm of path planning is inspired by the Dubins' theorem regarding shortest paths of bounded curvature in the absence of obstacles. The algorithm allows simplification of the problem model. The genetic algorithm is devised to search for the best path, and a special hierarchical structure of chromosome is used to denote a possible path in the environment. A penalty strategy is introduced to "punish" the infeasible chromosomes during searching. The performance results presented have shown that the approach is able to produce high quality solutions in several seconds.

Table of Content

Acknowledgements.....	II
Summary	III
Table of Content	V
List of Figures	X
List of Tables	XIII
<u>Chapter 1</u> Introduction	1
1.1 Motivations	1
1.2 Initiatives of RoboCup.....	2
1.3 Objectives	3
1.4 Major Contributions of the Thesis	4
1.5 Organization of the Thesis.....	5
<u>Chapter 2</u> System Architecture	7
2.1 Architecture of the Small Robot League	7
2.2 Tasks of the Vision System.....	9
2.3 Hardware and Software of the Vision System.....	11
2.4 Introduction to the AI module of the Small League	12
<u>Chapter 3</u> Vision System	15
3.1 Architecture of the Vision System.....	15
3.2 Image Processing Module.....	21

3.2.1 Color-based Segmentation	21
3.2.2 Blob Analysis.....	23
3.3 Tracking Module.....	24
3.3.1 Double Check Data Association	25
3.3.2 Velocity Computation.....	28
3.3.3 Local Marker Matching and Computation of Orientation	28
3.3.4 Elimination of Parallax Errors	29
3.3.5 Robot Identification	30
3.4 Experimental Results	37
3.4.1 Image Processing Steps.....	37
3.4.2 Tracking Static Objects.....	42
3.4.3 Tracking Moving Objects	44
 Chapter 4 Color and Camera Calibrations	 47
4.1 Features of Captured Image	47
4.2 K-Mean-clustering Color Calibration.....	49
4.2.1 Initial Class Center Selection.....	51
4.2.2 Color Sample Collection.....	51
4.2.3 Number of Classes in Clustering	52
4.2.4 Experiment Results and Discussion.....	52
4.3 Tsai's Camera Calibration Model.....	57
4.3.1 Camera Model.....	57
4.3.2 Transformation Steps	58

Table of Content

4.3.3 Acquisition of Matching Points	61
4.3.4 Result and Evaluation	64
Chapter 5 Introduction to Path Planning.....	67
5.1 Introduction.....	67
5.1.1 Dynamic Planning and Static Planning.....	68
5.1.2 The Basic Problem	70
5.1.3 Kinematic Constraints and Robot Model.....	71
5.1.4 C-Obstacles	73
5.2 Classical Algorithms.....	74
5.2.1 Grid-based Algorithms.....	74
5.2.2 Potential Fields Method	75
5.2.3 Roadmap Methods	76
5.3 Model of Static Path Planning in RoboSoccer.....	78
Chapter 6 A Path Planning Algorithm	81
6.1 Dubin's Theorem: Shortest Path without Obstacles	81
6.2 "Rope Solution": Shortest Path with Obstacles	84
6.3 Description of a Path.....	87
6.4 Searching and Genetic Algorithm.....	88
6.4.1 Relationships between a Path and a Circle	89
6.4.2 Complexity of Problem.....	90
6.4.3 Traveling Salesman Problem	90

6.4.4 Genetic Algorithm and TSP	91
6.4.5 Hierarchical Genetic Algorithm.....	92
6.5 Methodology	94
6.5.1 A Segment of a Path	94
6.5.2 Fitness Function	97
6.5.3 Criterion for Interception	97
6.5.4 Penalty Strategy	101
6.5.5 The Encoding of the Chromosome	102
6.5.6 Genetic operators	105
6.6 Simulation Results	107
 <u>Chapter 7</u> Conclusions and Future Research	 111
7.1 Contributions:	111
7.2 Recommendations for Future Research	113
(1) For Vision System.....	113
(2) For Motion Planning	115
 Author's Publications:	 120
Reference	121
Appendix A -- The Rules of Small Size Team	129
Appendix B -- Matrix DSP Card	140
Appendix C – DXC-9000 Camera with Zoom Lens	142
Appendix D – Data for DXC-9000 Calibration.....	143

Table of Content

Appendix E – Some TSP Operators..... 145

Appendix F. Linear Filter 149

List of Figures

NOTE: The image with (★) is in color, otherwise it is a grey scale picture.

Fig. 2-1 Robot soccer system	7
Fig. 2-2 Illustration of view of robot top-cover (★)	11
Fig. 2-3 RoboCup overall system in information flow	13
Fig. 3-1 The architecture of the vision system.	17
Fig. 3-2 Distribution of the colors in <i>RGB</i> space (★)	21
Fig. 3-3 Color-based segmentation on one interested color	22
Fig. 3-4 (a) Illustration of parallax error; (b) Removal of parallax error	30
Fig. 3-5 (a) The corners of the robot cover; (b) One additional color (★)	32
Fig. 3-6 The permutations of two colors for orientation and identification (★)	33
Fig. 3-7 (a) A general team formation; (b) Five patterns are selected in our team (★)	34
Fig. 3-8 (a) Image from first grab, (b) Image from second grab (★)	38
Fig. 3-9 Extracted image and its splitted <i>RGB</i> images. (★)	38
Fig. 3-10 (a) Objectless image. (b) Interested objects (★)	39
Fig. 3-11 The procedure to get the blob buffer of ball.	40
Fig. 3-12 Blob buffers of interested objects	41
Fig. 3-13 The client display (★)	42
Fig. 3-14 Evaluation of barrel distortion correction. (★)	43

List of Figures

Fig. 3-15 Tracking the objects with number tag. (★)	44
Fig. 3-16 Evaluation of tracking module. (★)	45
Fig. 3-17 Experiment result of tracking moving balls (★)	46
Fig. 4-1 Typical image for processing from a JAI 70 AC camera (★)	47
Fig. 4-2 Image used for clustering. (★)	52
Fig. 4-3 Image of clustering result with 8 classes (★)	53
Fig. 4-4 Image of clustering result with 10 classes (★)	55
Fig. 4-5 Class centers in color clustering with 10 classes	56
Fig. 4-6 Camera geometry with perspective projection	57
Fig. 4-7 The image of the calibration pattern	61
Fig. 4-8 Illustration of the matching algorithm	64
Fig. 5-1 The model of the robot and its constraints.	72
Fig. 5-2 C-obstacle	73
Fig. 5-3 Find a shortest route between A and B	75
Fig. 5-4 A visibility graph G	77
Fig. 5-5 A static model of the RoboSoccer field.	78
Fig. 6-1 The Dubin's car model	82
Fig. 6-2 Two types of optimal paths in Dubin's theorem	83
Fig. 6-3 (a) A random path like a rope; (b) A feasible path is like a tight rope	85
Fig. 6-4 The optimal path $\widehat{AB} + \widehat{BC} + \widehat{CF}$	86
Fig. 6-5 (a) Two ways around a circle; (b) The relationships between a path and a circle.	89

Fig. 6-6 Four possible types of segment of a path	95
Fig. 6-7 Two relationships of a line with a circle	98
Fig. 6-8 The criterion for the function $f(t)$ with two solutions t_1 and t_2	100
Fig. 6-9 The chromosome represents the path in Fig. 6-3	103
Fig. 6-10 The optimal path achieve from HGA.	107
Fig. 6-11 Fitness over generation	108
Fig. 6-12 A near-optimal solution	109
Fig. 6-13 Another optimal path generated by HGA	109
Fig. F-1 Linear filter model	149

List of Tables

Table 1-1 Comparison of chess and RoboCup	3
Table 2-1 Colors used in small robot league	9
Table 3-1 SRE and TDE in the case of PAL and NTSC format	16
Table 4-1 Result of K-Mean clustering with 8 classes	53
Table 4-2 Result of K-Mean clustering with 10 classes	54
Table 4-3 Camera Parameters of SONY DXC-9000	65
Table 4-4 The evaluation data by the full optimization of DXC-9000	66
Table 4-5 The evaluation data by the full optimization of JAI 70 AC	66
Table 6-1. Criterion for interception between a line and a circle	101
Table 6-2 The crossover and mutation operators	106

CHAPTER 1

Introduction

This chapter first introduces the Robot Soccer problem, including its motivations, objectives, and focus of current research. Then it outlines contributions of the current work, and organization of this thesis.

1.1 Motivations

The Robot World Cup Initiative or RoboCup is an international research and education initiative. It is an attempt to foster AI and intelligent robotics research by providing a standard problem where a wide range of technologies can be integrated and examined, as well as being used for integrated project-oriented education [RWCP02] [KTNO97].

In order for a robot team to actually play a soccer game, various technologies that must be incorporated include design principles of *autonomous agents*, *multi-agent collaboration*, *strategy acquisition*, *real-time reasoning*, *robotics*, and *sensor-fusion* [KTNO95] [KTNO97]. RoboCup is a task of controlling a team of multiple fast-moving robots in a dynamic environment. It also offers a software platform for research on the software aspects of RoboCup. Currently, RoboCup has the following leagues:

-
- Simulator League
 - Small Robot League
 - Full Set Small Robot League, which is 11 robots per team
 - Middle Size Robot League
 - Sony Legged Robot League (Sponsored by Sony)
 - Humanoid League (From 2002, demonstration may take place before 2002)
 - TeleOperation Track (to be announced)
 - RoboCup Commentator Exhibition

Details of the above are in [RWCP02].

1.2 Initiatives of RoboCup

The intentions of such a problem are to promote mobile robotics and AI research. In the case of RoboCup, the ultimate goal is to develop a robot soccer team which can beat a human world champion team. A more modest and achievable goal is to develop a robot soccer team which can play like human players. This is a formidable challenge, but accomplishment of such a goal has significant social impact. RoboCup is a landmark project as well as a standard problem. It may not be feasible with the current technologies of AI and other related areas, but with concerted efforts, the dream can become a reality. For example, in computer chess, the recent accomplishment by the *Deep Blue* team, which defeated *Garry Kasparov*, proves that AI can beat a human grand master by the official rules. The RoboCup could be another good problem to promote AI, to test various algorithms as the computer chess problem has done. Of course, the problem of RoboCup

Chapter 1 Introduction

is more diverse and difficult. Table 1-1 outlines the contrast between the RoboCup problem and the Computer Chess problem [OBRC02].

Table 1-1 Comparison of chess and RoboCup

	Chess	RoboCup
Environment	Static	Dynamic
State Change	Turns Taking	Real Time
Info. Accessibility	Complete	Incomplete
Sensor Readings	Symbolic	Non-symbolic
Control	Central	Distributed

The RoboCup offers an integrated research task covering the broad areas of AI and robotics. These areas includes *real-time sensor fusion, reactive behavior, strategy acquisition, learning, real-time planning, multi-agent systems, context recognition, vision, strategic decision-making, motor control, intelligent robot control*, and many more [KTNO95] [KTNO97] [OBRC02].

1.3 Objectives

The work done in this thesis is for application in the small robot league. In this small robot league, there are many tasks to be accomplished, and the vision module is identified to be an important module. One of the main objectives of this thesis is to build a vision module for tracking the current states of our robots, opponents and balls. Camera and

color calibration are also very important for the vision module for they provide the parameters for tracking.

Another challenging problem in robot soccer is motion planning. Few researchers have tried to find an optimal path in this difficult field. The path planning problem can be divided into dynamic motion planning and static path planning. The dynamic motion planning will be very useful for robot soccer but it is considered to be a very difficult problem. A modest objective of this thesis is to develop a solution to the path planning problem in a static environment.

1.4 Major Contributions of the Thesis

There are three main contributions in the thesis.

1. A robust real-time tracking system for the RoboCup small robot league. The tracking system tracks and predicts the motion states of the robots and a ball in the field and sends information to the AI module. The tracking module locates the features by low-level image processing, it then associates the current states with previous states and updates the current states, and finally, it sends information to the AI module.
2. Color and camera calibration are implemented to provide the parameters for the tracking module. The classification is used here to provide the color parameters for segmenting the image. Tsai's camera calibration model is used here to provide the camera parameters for tracking.

Chapter 1 Introduction

3. A new hierarchical genetic algorithm (**HGA**) for path planning in a static environment with obstacles. The nonholonomic constraint dramatically changes the motion of the robot. The algorithm is developed by combining the Travelling Salesman Problem (**TSP**) by Genetic Algorithm and Dubins' theorem regarding shortest paths of bounded curvature in the absence of obstacles.

1.5 Organization of the Thesis

The thesis is organized as follows:

Chapter 1 gives the introduction to the research topics. Background information and objectives of the research are discussed.

Chapter 2 introduces the general information of the RoboCup small robot league, including the architecture of the small robot league, the vision system and its tasks, and the implementation of the software and hardware of the system. An introduction on the AI module is also given to provide better understanding of the system.

Chapter 3 provides a robust design and implementation of the vision system. Vision System plays a very important role in the small robot league. As the eyes to the robots, it plays the role of the sensor to the whole system.

Chapter 4 introduces two kinds of calibrations in the vision system: color calibration and camera calibration, which are also very important in the vision system.

Chapter 5 introduces several classic algorithms for path planning. They include *Distance Transform*, *Lee's algorithm*, *Artificial Potential Fields* algorithm, and *Roadmap Method*. We will introduce our problem arising from the mobile robots model.

Chapter 6 develops a novel path planning algorithm based on a genetic algorithm. First, we will introduce Dubin's algorithm, which provides us with the shortest path without obstacles. Then we will introduce the main idea of our algorithm, which is called "rope solution" because the path is like a rope in the environment with obstacles. However, it is not promised to provide an optimal path. The main idea of the algorithm is about searching, so a genetic algorithm is used in this problem.

Chapter 7 evaluates the research and contributions on the tracking system, calibration, and path planning and then discusses the directions for future research in this challenging problem.

CHAPTER 2

System Architecture

This chapter introduces the architecture of the small robot league, the vision system and its tasks, and the implementation of the software and hardware of the system. It also includes an introduction on the AI module for robot soccer system.

2.1 Architecture of the Small Robot League

Fig. 2-1 Robot soccer system

The robotic team is an autonomous system consisting of a preset processing cycle (see Fig. 2-1). The system architecture of the robot soccer team is shown in Fig.2-1. The vision module receives the raw image sequence from a camera mounted on top of the field and processes the information captured, giving the motion states of the ball and each robot. This information is sent to the intelligent command module, which evaluates the world state and decides what to do in each step based on its specific strategy. Robots' actions are motion commands broadcasting by the control module through wireless communication.

Most teams today adopt the global vision system as seen in Fig. 2-1 [NOKH00] [BOWL00]. Why is the global vision system adopted in the vision system? The 12th rule of small size team is about which vision system should be used, it can be a global vision system or an external distributed vision system. According to this rule, a team can select one global camera vision system, or a combination of a few cameras [RGLT02]. At the first glance, more cameras will be better for a vision system, but most teams today use a global vision system. The most important reason for adopting the global vision system is that it is sufficient for the small size team. A minor reason is that although more cameras provide more information, it is not so easy to implement the vision system because of limit on size, weight and energy. However the distributed vision system is adopted in other leagues [OSWD00] [MORN00].

It is very important that the vision system is robust. In our system, the global vision is the only way to collect information provided for high-level control. It means that if the vision

Chapter 2 System Architecture

module cannot work properly, the whole system will have a poor performance and can even crash.

2.2 Tasks of the Vision System

The objective of the vision system is to extract useful information from the environment. In the RoboCup Small Robot League, in order to make the image processing and pattern recognition feasible for real-time usage, several important constraints are adopted [RGLT02]:

Table 2-1 Colors used in small robot league

Color	Representation of the object
Orange	Ball
Yellow	One team
Blue	The other team
Green	Playing field
White	Field markings including border, middle line, center circle, and defence zone lines
Black	Physical robot's cover
Magenta	Orientation Mark(Optional)
Cyan	Orientation Mark(Optional)

- Lighting: 700-1000 LUX uniform light is intended, but not guaranteed.

- Playing field: Dimensions of the playing surface are 152.5 cm by 274 cm. The floor material is green felt mat or carpet.
- Color: Six main colors are used to differentiate objects (see Table 2-1).

See Appendix A for important rules and regulations.

The first question is, “Which kind of information should be provided by the Vision System to the Intelligent System?”

The information of motion states of the moving objects is provided for high-level decisions, which can be different for different requirements. Generally, robots’ positions and identifications (ID) are indispensable to their representation. Besides these two data, robots’ directions and velocities are also included in our system. A robot’s motion state is therefore defined as a vector: $[ID, x_r, y_r, \theta, v_{rx}, v_{ry}, \omega]^T$, where ID is its identification; (x_r, y_r) is its position; θ is its direction; ω is rotational velocity, and (v_{rx}, v_{ry}) is the velocity of the robot. Similarly, the ball’s motion state is defined as $[x_b, y_b, v_{bx}, v_{by}]^T$, here (x_b, y_b) is the position of the ball, (v_{bx}, v_{by}) is its translational velocity.

The main objective of our vision system is to track 11 moving objects (one ball and ten robots) in real time. There is a distinct requirement of computation for each element in the motion vector. The retrieval of robot’s position and direction takes up most of the time since they are obtained by processing the raw data of image sequence. A yellow or blue ping-pong ball, which is fixed on the robot top-cover, is used to signify a robot’s position belonging to a team. In order to obtain the robot’s orientation, extra markers are

Chapter 2 System Architecture

needed (see Fig. 2-2). So, the image processing results are blob centers of the ping-pong balls and the extra markers. Robots' ID and velocities are extracted based on the center data of two sequential frames.

Fig. 2-2 Illustration of view of robot top-cover

2.3 Hardware and Software of the Vision System

To fulfill the tasks of the vision system, the design of the vision system should be provided first, then the configuration of details of software and hardware. Although this part is related to the next chapter's analysis, a brief introduction on the configuration here will make the description clearer.

The major hardware of the vision system is somewhat standard. Normally in a real-time inspection system, the CCD camera is a must to capture the image. Often, a DSP board is used to make the feature extraction in real-time. Hardware of the vision system comprises the following:

- Lighting: 700 – 1000 LUX uniform intensity is expected but not guaranteed.
- Playing field: A table tennis field (152.5 cm x 274 cm) covered with green colour carpet or mat.
- The Mobile Robot's back cover, and ball.

-
- SONY DXC-9000 CCD progressive scan full frame colour camera or *CV-M70* CCD cameras.
 - A camera pole to position the camera above the playing field.
 - Matrox Genesis Image Processing Board with 64 MB of SRAM and Grab and Display Modules.

Useful hardware details are introduced in the Appendices B and C.

The processing steps executed in the software of the vision system comprises:

- Image acquisition
- Color-based image Segmentation
- Blob Analysis and locating objects
- Object association within two consecutive images
- Computation of velocity and orientation of objects
- Tracking of objects with linear filters
- Data transmission using UDP to the AI system.

The details of the processing steps will be introduced in the next chapter.

2.4 Introduction to the AI module of the Small League

Fig. 2-3 demonstrates the feedback information flow of the overall RoboCup system. It is flexible to design the AI module. That means different teams can adopt different strategies. The small robot soccer system can be divided into two independent modules, and our research in this thesis is mainly concern with the vision system. However the

Chapter 2 System Architecture

introduction to the AI module of some successful teams will be helpful in understanding the small robot league.

Fig. 2-3 RoboCup overall system in information flow

CMUnited-98 was the champion in 1998. In the team’s architecture, the AI module is divided into a main module and 5 client sub-modules. After the main module receives the state information in the field, the main module sends the object position information to each client module. The client module after calculation sends action codes to the main module, which transmits the commands to the physical robot using radio communication. Obviously, in this model, the client module decides which action to do and controls the real-time robot’s motion after action is selected [BOWL00]. Big Red, a team from Cornell University, was the champion in 1999. This team pays much more attention to feedback control of the robot than to its AI aspects. This team can generate a good trajectory with obstacle avoidance for the robot. And role-based strategy is another feature applied in Big Red [ANDR00].

Machine learning algorithms are also commonly used in robotic soccer teams. These algorithms select a hypothesis from a hypothesis space based on a set of training examples such that the chosen hypothesis is predicted to characterize unseen examples as accurately as possible. Each hypothesis maps a set of input features to a set of output features. The layered learning algorithm of the CMU simulated robotic soccer team is a paradigm of machine learning applications. Since it is too complex for learning directly from the input to the output representation, the problem is divided into three layers, for example, ball interception for layer one, pass evaluation for layer two, and pass selection for layer three. For each layer, different machine learning algorithm is adopted. The team with layered learning is the champion at RoboCup-98 and RoboCup-99 [STON98] [STON00]. This model is too complex to be evaluated quantitatively.

From the reviews of the different teams' designs and implementations, there can be totally different ways to achieve the common objective. Because of the complexity of the robotic soccer, it is not difficult to find that at the present moment, practically soccer team's designs are still not so elegant and hard to evaluate. And in the long run, there will be great processes in the robotic soccer small robot league, be it in vision, AI, control, and/or motion planning.

CHAPTER 3

Vision System

The vision system of robot soccer plays a very important role in the small robot league. Without precise and stable state information of the field, the AI module would not know what to do. As the eyes of the players, the vision system plays the role of the sensor to the whole system. In this chapter, we provide a robust design and implementation of a vision system.

3.1 Architecture of the Vision System

Efficiency and *robustness* are two crucial requirements for the vision system. “*Efficiency*” here means the vision module can track the moving objects at least at the standard frame rates, so that it can provide the real-time motion states of objects to the high-level decision module with little or no delay. “*Robustness*” means the vision system can cope with exceptions that might occur while tracking. The typical exception is the loss of targets, which can be due to the unexpected high speed of objects or inaccurate color calibration.

To the vision system, processing speed and accuracy are often two contradicting factors, because faster processing speed implies simple algorithms that often lead to coarse results. Therefore, we must address the following problem first before determining the vision system architecture:

“Which is more important, processing speed or accuracy?”

Notations for Two Kinds of Errors.

Spatial Resolution Error (**SRE**) - The error caused by the image resolution.

Temporal Delay Error (**TDE**) - The error caused by the sampling speed and objects moving speed.

In fact, there are two basic problems inherent in our system that are caused by camera resolution and perception sampling rate (this rate is proportional to the processing speed.): Spatial Resolution Error (**SRE**) and Temporal Delay Error (**TDE**). Let us examine SRE and TDE on the condition that the camera supports two standard video formats, **PAL** (25fps, 720x576) and **NTSC** (30fps, 640x480).

Table 3-1 SRE and TDE in the case of PAL and NTSC format

Errors	SRE (m)		TDE (m) at 2 m/s
	Along the length (2.74 m)	Along the width (1.525 m)	
PAL	$2.74/720 = 0.0038$	$1.525/576 = 0.0026$	$2/25 = 0.080$
NTSC	$2.74/640 = 0.0043$	$1.525/480 = 0.0032$	$2/30 = 0.067$

Chapter 3 Vision System

Suppose that the maximum linear speed of a moving object is no less than 2 m/s. This assumption is reasonable since it was reported that the robots of many teams could attain this speed [ACHM96] [WYTH99] [ANDR00].

For example, if PAL format is selected. Since the image size is 720x576, and the field size of robotic soccer is 2.74x1.525, the SRE in one direction is $2.74/720$ m, and in another direction is $1.525/576$. Since the camera provides 25 frames per second, and the maximum speed of the object is about 2 m/s, the TDE is $2/25$ m. From Table 3-1, it is found out that TDE is about 20 times that of SRE, which means SRE is negligible compared with TDE and high processing speed will improve system's performance greatly. So it is important to accelerate the processing speed. However, high processing speed does not come free. It needs powerful hardware and proper computation resource assignment to provide the necessary support.

Fig. 3-1 The architecture of the vision system.

As shown in Fig 3-1, the vision system is divided into two parts: a special DSP vision card and the host computer. The lower module such as blob analysis should provide the original position of the objects captured in the image. The higher module such as linear filter should estimate and filter the data according to the tracking algorithm.

The objective of vision system is to track the state information of the robots and ball. The lower module of the vision system is basically an image processing module. To achieve high processing speed, normally this module is developed with the powerful hardware like DSP or FPGA card. There may be more than one way to achieve the lower module. However, from analyses of the properties of the captured image from the camera and the rules of the RoboCup in Appendix A, it is found that color is the most important feature in image processing. Colors in the small robot league are almost independently separated around the corners of the RGB space, and they can be differentiated in the RGB space though other teams may use the HSI space [SIMN01]. Blob analysis is based on image segmentation and provides the blobs' feature information, normally the center of gravity [GNZL92] [GNLC00]. So it is natural to choose the color-based segmentation and blob analysis, which is an excellent way to implement the lower module.

After getting the raw position data from the DSP board, the host computer completes the computation of velocities, ID matching and data transmission. Since the velocity obtained by differentiating position is very sensitive to the errors of position, filtering techniques are also applied to suppress the noise and provide more precise state information by

Chapter 3 Vision System

prediction and update [BRKN88]. For the higher module, Kalman filtering is a very classic method for tracking but it is quite complex, and its initial parameters cannot be obtained from experiment. Thus a linear filter is adopted for its simple form and efficiency.

The output of the vision system, i.e. the objects' motion states, is sent to the intelligent module via an independent LAN. UDP protocol is selected for real-time information without the overhead of TCP protocol. LAN is separated from other networks, which avoids most of general network problems, such as traffic jam, and dropped packets [TNBM96][STVN94].

The above system architecture makes full use of the power of the vision card and PC, so that it optimizes the processing speed. An additional advantage is that the vision system is separated from other functional modules, which minimizes the possibility of interferences and makes it easy to update each module without affecting others.

The function of each step is briefly stated as follows:

1 Color-based Segmentation, whose objective is to achieve the binary buffers of the interested colors from the captured image (done on the DSP board).

2 Blob Analysis, whose function is to calculate the position of each blob from the binary buffers, and filter out unwanted noise (done on the DSP board).

3 Data Association, whose main function is to associate the previous states with the new states from the blob analysis (done on the host computer).

4 Velocity Computation, whose objective is to filter the translational velocity of the robots and the ball (done on the host computer).

5 Local Mark Matching, whose objective is to estimate and filter the orientation and rotational velocity of the robots (done on the host computer).

6 Elimination of Parallax Errors, whose main objective is to eliminate one type of error in the vision system (done on computer).

7 Robot Identification, whose main function is to identify the robots (done on computer).

These steps need not be processed exactly in the order mentioned above. In real-time tracking, the order of some of the steps can be swapped. The following sections will give the details of each step.

3.2 Image Processing Module

3.2.1 Color-based Segmentation

Fig. 3-2 Distribution of the colors in RGB space

Image segmentation is a low-level image processing task that aims at partitioning an image into homogeneous regions [DUDA73]. Fig. 3-2 shows the distribution of the colors used in the experiment in RGB space. It is natural to apply the threshold to segment the image.

Fig. 3-3 Color-based segmentation on one interested color

Fig. 3-3 shows the algorithm for color-based segmentation. For a specific RGB color, there are 6 parameters we should know before segmentation, and these include the minimum and maximum threshold values of each of the 3 bands. If the color distribution in the real environment is identical to its ideal case, then color threshold selection is a trivial matter. Unfortunately it is not true because of lighting and noise. The selection the color parameters can be done manually, but it will take a long time. In chapter 4, a color calibration algorithm is given and we can see the distribution is not so good.

Threshold operation is defined as:

$$\text{Threshold}(x, \min, \max) = \begin{cases} 1 & \text{if } (\min < x < \max) \\ 0 & \text{else} \end{cases} \quad (3.1)$$

Here x is the intensity of a certain pixel in a band buffer.

Chapter 3 Vision System

After threshold, each color band buffer becomes a binary buffer. Then logical AND is performed on the same position of three binary buffers. The result of logical AND is a binary buffer for a blob with an interested color, that means if the bit in that position is '1', there is at least one interested blob around the position. Each interested color corresponds to one buffer, so after this step there should be one buffer for ball, two buffers for two teams' marks, and may provide optional buffers for orientation marks.

3.2.2 Blob Analysis

Blob analysis is very common used in tracking [HOCH00]. The term blob refers to a region of the image where all of the pixels in a 4-connected or 8-connected region are of a homogeneous value. Blob analysis is a branch of image analysis that allows you to identify connected regions of pixels (blobs) within an image, and then to calculate selected features of those regions. Typical features describe the size, shape, and location of the blobs. Therefore, one can use the results of blob analysis for a variety of purposes, for example, to distinguish between objects in an image, to locate objects, and to measure objects. Along a production line, for instance, one can use the results to determine if parts have been manufactured within specified tolerances [GNZL92] [GNLC00].

This section discusses some of the blob features supported by the Genesis Native Library [GNLC00]. Most features fall into one of four main groups: area and perimeter, dimensions, shape and location.

The following are the equations for blob analysis:

$$cente_of_gravity_x = \sum_{i \in Blob} x_i / number_of_pixel_in_Blob \quad (3.2)$$

$$cente_of_gravity_y = \sum_{i \in Blob} y_i / number_of_pixel_in_Blob \quad (3.3)$$

It is also the center of gravity equation.

The center of the blob is the location of the object in the image. And the areas and the perimeters are used to filter out unwanted noise. After blob analysis, the positions of interested color are known.

Blob analysis takes much more time than other operations, but there are ways to make it faster. In fact, there is a lot of redundant information between the consecutive frames. Compared with the previous frame, the new frame only provides a little additional useful information on the objects in the field. Using filtering techniques, we can produce a small predicted region where the object might be in the next step. But in the DSP board, it will take a lot of overhead to extract so many regions from the binary buffers. A team adopting such a technique can analyze 30 frames per second on a standard PC without the DSP board [SIMN01].

3.3 Tracking Module

The objective of the tracking module is to provide the precise state information of the objects in the field and to handle some exceptions from image processing.

Chapter 3 Vision System

3.3.1 Double Check Data Association

Data association is an important problem in multiple target tracking. After obtaining the current frame blobs information, data association is set up to map relationships between the blobs in the current frame and the blobs in the previous frame. It is necessary to associate each detected robot in the field with its identification in order to control it. If the image processing speed is low, it is highly possible that there will be errors due to the overlap of fast moving robots. Actually if the processing speed is fast enough, for example 25 frames per second, it can cope with a maximum speed at 2m/s, so it is normal for a robot to have 8 cm radius circular region. If there are two robots in this region, a wrong association may occur.

CMUnited-97 provided a greedy algorithm to fulfill data association, which is maintained by searching for objects within a minimum displacement. That means one blob in the current frame is only associated with the blob in the previous frame of the minimum distance. Due to noise, it is possible for the detection system to leave a robot undetected. Another technique that is commonly used is to make the addition of the distance between the associated pair blobs minimal. This way is better than the greedy association algorithm but requires more computational time [VLSO98].

The following algorithm in this thesis is based on the principle of the greedy association algorithm, but improved by scanning two times. This algorithm uses a parameter named **MaxDistBetweenFrame**, that means if the distance of the association objects is less than the parameter, the association is success with a high probability; if the distance is larger

than the parameter, the second time scan associates the remaining unmatched objects greedily with a minimum distance. This algorithm can avoid wrong association during the first scan. In fact, if the prediction is adopted, the result will be better. That means the current blobs can be associated with the predicted blobs position.

The following algorithm is based on two sequential frames. A double-scan algorithm is applied. Its pseudo code is as follows:

Double scan mapping algorithm

```

n is the robots number in the field;
    assume that the number of robots don't change;
Let Pred[1..n] be the array of blobs in the previous frame
Let Curr[1..m] be the array of blobs in the current frame
Let MaxDistBetweenFrame be the maximum distance of a robot in the two
    sequential frames.
Let DistPairs(i,j) be Dist(Pred[i], Curr[j]), that is, the distance
    between Pred[i] and Curr[j].

First scan: Match the blobs in the two sequential frames according to
    minimum distance principle on the condition that this distance must be
    less than or equal to MaxDistBetweenFrame.
if (n <= m){ // Current blobs # <= Previous blobs #
    // Select n blobs in the current frame
    // to match the blobs in the previous frame
    for(i = 0; i<n; i++){
        j = argmin{DistPair(i,l) < MaxDistBetweenFrame}, l = 1..m;
        if (j exists)
            Save the matching result;
            Set row i and column j of {DistPair(i, j)} to be infinity;
    }
}
else { // Current blobs # (m) > Previous blobs # (n) (some noise here)
    // Select m balls in the current frame to match the blobs
    for (j = 0; j < m; j++){

```


Chapter 3 Vision System

```

        i = argmin{DistPair(l,j) < MaxDistBetweenFrame}, l = 1..n;
        if (i exists)
            Save the matching result;
            Set row i and column j of {DistPair(i, j)} to be infinity;
    }
}

```

Second Scan: Match the remaining blobs out of the scope of MaxDistBetweenFrame. This is an exception handler.

```

if(n <= m){
    // Select n blobs in the current frame
    // to match the blobs in the previous frame
    for(i = 0; i<n; i++) {
        //search matching results after first scan along the row
        if(ith element has not been matched) {
            j = argmin{DistPair(i,l)}, l = 1..m;
            if (j exist) {
                Save the matching result;
                Set all elements in row i and column j to be Infinity
            }
        }
    }
}
else{
    // Select m balls in the current frame to match the blobs
    for(j = 0; j < m; j++) {
        //search matching results after first scan along the column
        if(jth element has not been matched) {
            i = argmin{DistPair(l,j)}, l = 1..n;
            if (i exists){
                Save the matching result;
                Set all element in row i and column j to be infinity;
            }
        }
    }
}
}

```

3.3.2 Velocity Computation

The mapping algorithm connects the blobs in the current frame with those of the previous frame. Based on the positions of two matched blobs, the velocity can be obtained as

$$v_x(i) = \frac{x(i) - x(i-1)}{T} \quad (3.4)$$

$$v_y(i) = \frac{y(i) - y(i-1)}{T} \quad (3.5)$$

where T is the time difference between two frames. Since T is generally very small, the computation of v is fairly sensitive to the detection errors in (x, y) . Our system therefore apply $g-h$ linear filter (*prediction update equation*) to make the estimation. It suppresses the affect of random errors. See Appendix F for more information about the linear filter.

3.3.3 Local Marker Matching and Computation of Orientation

Because the size of the robot's top cover is known, it is easy to define a local region around the robot's position to look for extra markers. This process is called local orientation marker matching. If successful, assuming the position of the extra marker is (x_e, y_e) , the orientation θ of a robot is

$$\theta(i) = \tan^{-1} \left(\frac{y(i) - y_e(i)}{x(i) - x_e(i)} \right) \quad (3.6)$$

Chapter 3 Vision System

where (x, y) is the position of the robot. If local matching is unsuccessful, the last motion commands and the orientation values are used to estimate the direction of the robot in the current frame, that is,

$$\omega(i) = \frac{\theta(i) - \theta(i-1)}{T} \quad (3.7)$$

where ω is the rotation speed of the robot, which can be worked out from the speeds of left and right wheels. Our system also applies $g-h$ linear filter in the prediction of the translational velocity.

3.3.4 Elimination of Parallax Errors

There are two kinds of errors in spatial position: barrel distortion and parallax errors. We discuss the barrel distortion and how to eliminate it in the next chapter, while the parallax errors is quite easy to solve as shown in Fig. 3-4, because it is caused by the height of the object, the height of the camera, the relative position of the object to the camera center in the world coordinate, if assuming the camera is perpendicular to the field. The following relationship is achieved:

$$\frac{x' - x}{x' - x_0} = \frac{h}{H} \quad (3.8)$$

$$\frac{y' - y}{y' - y_0} = \frac{h}{H} \quad (3.9)$$

Fig. 3-4 (a) Illustration of parallax error; (b) Removal of parallax error.

The symbols in the above equation are shown in the Fig. 3-4 (b). So the parallax errors can be removed by the following functions:

$$x = x' - \frac{h}{H}(x' - x_0) \quad (3.10)$$

$$y = y' - \frac{h}{H}(y' - y_0) \quad (3.11)$$

3.3.5 Robot Identification

The artificial intelligence module sends the commands to each robot with the format like **<robot ID, speed of the left motor, speed of the right motor>**, here 'robot ID' represents the physical identification of the robot which executes the commands only with the same ID as itself set in the program. There is one ID for each robot in the AI system and one corresponding ID in the vision system. Also, the ID for each robot in the field should map to the ID in both the AI and vision system. The identification of the

Chapter 3 Vision System

robot is through the relationship of the mapping. If there is not such mapping relationship, the system may not control the robots properly.

Let us consider one condition. If each robot of our team is homogenous in color and shape, then the vision system cannot tell the difference between the robots. Since the AI module itself is unable to differentiate the robots, it may send the control command meant for one robot to another robot. In such a case, the whole system cannot work properly.

Maybe the AI module can differentiate the robot by comparing the actions with the command. However it has several deficiencies: It is difficult to identify the robot within the first several cycles. The data association is not 100% right, so there may have some errors. The recovery from the fault association will take time. It may cause large time delay.

The reason for this difficulty lies with the use of homogenous robots, and the artificial intelligence module cannot overcome this problem.

3.3.5.1 Design of the role pattern

Fortunately, we can overcome this problem in the computer vision module, but we must create inhomogeneous robots, that is, robots with different orientation marks in shape or color.

Generally, the robot is assumed to have a rectangular shaped cover. We decide to use the four corners of the rectangle to identify the orientation and the identification of the robot. These two kinds of information are indispensable for the motion control of the robot. At the same time, for the efficiency of real-time tracking, one cannot use too many colors.

Fig. 3-5 (a) The corners of the robot cover, (b) One additional color

There are four corners in the cover of the robot as shown in Fig 3-5 (a). These four corners on the cover of the robot can be used for orientation and identification. First we want to know if one additional color is enough for 5 robots in a team. In Fig. 3-5 (b), only one color (cyan) is considered for orientation and identification. The figure shows that the number of permutations for this color is five, the same number as that of the robots. Only the 3 results of 1st row have the orientation, while the 2 covers on the 2nd rows have no orientation because of symmetry. So only one color is not enough to identify all five robots for one team, additional colors must be adopted.

Chapter 3 Vision System

Fig. 3-6 The permutation of two colors for orientation and identification.

Fortunately, we only need two additional colors. Fig. 3-6 shows all the permutation which can identify the orientation of the robots of a team. So five patterns must be selected from among all these 12 permutations. We want to select 5 patterns, which can be identified for all the robots of a team.

First, let us think about a team formation, which is shown in Fig. 3-7 (a). The LF (Left Forward) and RF (right forward) are mostly in the front portion of the field, the MF (middle field) and DF (Defender) spend most of their time on the middle and back portion, and GK (Goal Keeper) is always around the goal zone for its duty. Sometimes, these roles may overlap, but for most of the time they will keep the formation because in this way, the team can use less energy, at the same time they can fulfill the same goal. Hence,

1. LF must be identified from RF, MF, and DF;
2. RF must be identified from LF, MF, and DF;
3. MF should be identified from any other robot, so should DF;
4. and GK should be identified from MF and DF.

Fig. 3-7 (a) A general team formation; (b) Five patterns are selected in our team

Although all the 12 patterns we designed are different from each other, it is important to choose patterns that have the greatest difference, which can reduce the possibility of wrong identification. So we should choose the patterns which are distinctly different from each other, especially for the roles which are close to each other in the formation of the field. Fig. 3-7 (b) shows the selected patterns to identify the orientation and the identity of the robots. The patterns are chosen according to the principle mentioned above.

3.3.5.2 Identification of the Roles

1 The main procedure

Chapter 3 Vision System

The aims of robot identification are to identify the role of a robot and calculate its orientation, to correct some errors of data association and calculate its orientation. After data association, the current blobs of the team marks are mapped to the previous team marks. We can search for the orientation marks around the team marks, and calculate the orientation of the robot. The following are the pseudo codes for the main procedure of identification.

```
// If our team, identify the role and calculate the orientation
Identify the GK, calculate the orientation;
Identify the DF, calculate the orientation;
Identify the MF, calculate the orientation;
Identify the LF, calculate the orientation;
Identify the RF, calculate the orientation;
```

2 Role Identification

The basic idea for identifying a role is to use the basic calculation of complex variables. The following are the pseudo codes for the identification of GK, the codes for other roles are omitted as they are similar.

Identification of GoalKeeper

```
// ◆ means mark "1".    ■ means mark "2".    ● means team mark.
// The orientation marks are arranged like following pattern
// around the team mark.
// ◆ ↑ ■
//    ●      This pattern means GK.
// ◆   ■
// Calculate the orientation based on data association
```

```

if(the position of the GK is available after data association)
{
    Search the orientation marks around the team blobs,
    at most four vectors are available, Z1 and Z2 for orient1,
    Z3 and Z4 for orient2;
    if((there are 2 "1" and 2 "2" around)    // exactly match
    ||( there are 2 "1" and 1 "2" around)    // partially match
    ||( there are 1 "1" and 2 "2" around)){ // partially match
        calculate the orientation using the complex variable;
        return;
    }
    else{
        //The result based on the team mark is unreliable;
    }
}
// If the result based on the team mark is unreliable
// or if the data of the data association is not available,
// search all the team marks for the identification.
// This is an exception handler.
for(each blob available){
    Search the orientation marks around the blob,
    at most four vectors are available, Z1 and Z2 for orient1,
    Z3 and Z4 for orient2;
    if((there are 2 "1" and 2 "2" around)    // exactly match
    ||( there are 2 "1" and 1 "2" around)    // partially match
    ||( there are 1 "1" and 2 "2" around)){ // partially match
        calculate the orientation using the complex variable;
        if( the role of the blob != the GK)
            swap these two blobs
        return;
    }
}
If still can't identification the role, maybe the robot is lost for
some reason.

```

There are 2 steps in each function, the first step is to identify the robot according to the result of data association. If this step fails, the second step will take place. The second

Chapter 3 Vision System

step is to search all the team blobs for the current role, calculate the orientation if the role is found, and sort the order of the team blobs.

3.4 Experimental Results

We have made experiments on *CV-M70* CCD camera and *DXC-9000* CCD camera. *CV-M70* CCD progressive scan full frame color camera was chosen in the following experiment. With a resolution of 450 x 480 (TV lines) and SNR at 48 dB, the camera is able to update images at a rate of 60 frames per second.

We first show the procedure of vision module. To simplify the results, we only use one orientation mark for a robot, but the principle is same. Later we will show the result of tracking moving objects.

3.4.1 Image Processing Steps

a) Image capture

Fig 3-8 shows the two frames from first two grabs. The image format is 320 x 240. There are 4 colors of interest: orange (ball), yellow (team mark), blue (team mark), magenta (additional mark). In this example, only one orientation mark is considered.

Fig. 3-8 (a) Image from first grab, (b) Image from second grab

b) Split the image into RGB buffers

Fig. 3-9 Extracted image and its split *RGB* images.

Chapter 3 Vision System

Fig. 3-9 shows the extracted image from the captured image and its splitted *RGB* bands:

(a) The extracted image with 296 x 160 pixels from Fig. 3-8, (b) red band image, (c) green band image, and (d) blue band image.

c) Using objectless image (Optional)

This step is optional, but it is helpful to have this step. At the beginning, an objectless image is captured and saved, as shown in Fig. 3-10 (a). First, the captured image subtracts the objectless image, then a binary buffer or mask buffer is achieved after threshold. The dark domain in the mask buffer means there are no objects of interest there, while the remaining domain of the mask buffer means there are objects of interest there. After the mask buffer is achieved, the objects of interest as shown in Fig. 3-10 (b) can be selected from the extracted image from Fig. 3-9 (a).

(a)

(b)

Fig. 3-10 (a) Objectless image. (b) Objects of interest

d) Get the blob buffers

Fig. 3-11 shows the procedure to get the binary blob buffer of a ball. The point in the ball can be describe like

$$\{P(r,g,b) | P \in Ball, r \in [r_{min}, r_{max}], g \in [g_{min}, g_{max}], b \in [b_{min}, b_{max}]\}.$$

Here P is a point.

Fig. 3-11 The procedure to get the blob buffer of ball.

There will be red color threshold in the red buffer, green color threshold in the green buffer, and blue color threshold in the blue buffer. After thresholding, there will be three binary buffers, as shown in Fig. 3-11(a) the red band image after color thresholding

Chapter 3 Vision System

(Ball), (b) the green band image after color thresholding (Ball) and (c) the blue band image after color thresholding (Ball). Fig. 3-11 (d), which is blob identifier image for balls, is achieved from the *AND* operation of Fig. 3-11 (a), (b) and (c).

For other objects, like team A mark (blue), team B mark (yellow), orientation mark (magenta), the blob buffers can also be achieved using the same way. Blob analysis result is omitted here because it is very obvious.

Fig. 3-12 Blob buffers of objects

Fig. 3-12 shows blob buffers of objects: (a) shows the blob identifier image for balls, (b) shows the blob identifier image for orientation markers, (c) shows the blob identifier image for team A markers, and (d) shows the blob identifier image for Team B markers.

After blob analysis, Tsai’s model is adopted to correct the barrel distortion. From this point onwards, the results shown are compensated using the Tsai’s model [TSAI86] [TSAI87].

3.4.2 Tracking Static Objects

A client display module as shown in Fig. 3-13 is developed to analyze the effectiveness of the vision module, including the distortion correction.

Fig. 3-13 The client display

The client display is running on a separate station (AI system). The module allows the user to select the quantity of balls and robots to be tracked. “Thy” denotes one team’s

Chapter 3 Vision System

robot and “Them” denotes another team’s robots. Since the module can be run on any computer in the LAN, the module requires the user to key in the Internet Protocol (IP) address of the vision system. The green grid lines are used to indicate the locations of objects effectively.

To evaluate the performance of distortion correction, a ball and two robots covers (the team A and B markers with orientation markers) are placed at three corners as illustrated in Fig. 3-14 (a). The display illustrates the locations of each object clearly in Fig. 3-14 (b) with the removal of the barrel distortion.

Fig. 3-14 Evaluation of barrel distortion correction.

The result reflected in Fig 3-14 (b) shows that the ball, the team A and B markers and the orientation markers are located correctly. Evaluating the result using the grid lines in the display, it is shown that barrel distortion in the captured image is compensated. In fact, the difference of real measurement the object position and estimation is less than 1 cm. The result is very good considering that one pixel in the image is about 1 cm in world coordinates.

In another experiment, the objects to be tracked were increased to five balls, four team A markers and five team B markers including orientation markers. Fig. 3-15 (a) and (b) illustrate the captured image and the result on display, respectively.

Fig. 3-15 Tracking the objects with number tag.

3.4.3 Tracking Moving Objects

To evaluate the tracking performance, a ball was pushed horizontally across the field. A manual simulation is shown using the Fig. 3-16 below.

Chapter 3 Vision System

Fig. 3-16 Evaluation of tracking module.

In Fig. 3-17, five orange identical balls are arranged at different positions on the playing field. One ball is pushed and its motion is tracked for a 1 second duration. 30 frames are recorded, one of which are presented in Fig. 3-17 to show the objects' motion. The number near the block is an ID number associated to an object.

Fig. 3-17 Experiment result of tracking moving balls

The tracking performance is good enough to track an object. Numerous experiments were done to determine the velocity that the tracking module is able to track. The result of the experiments showed that the tracking module does not encounter any difficulty when tracking a ball with a maximum velocity of 5m/s.

CHAPTER 4

Color Calibration and Camera Calibration

This chapter introduces color calibration and camera calibration in the vision system.

4.1 Features of Captured Image

Fig. 4-1 Typical image for processing from a JAI 70 AC camera

Fig. 4-1 shows a typical image for processing by the JAI 70 AC camera. There are two main features in the vision system.

Since color-based object detection is adopted for the vision system, accurate color thresholds are also required. From Fig. 4-1, one can easily observe the following:

- There are many weak yellow pixels scattered on the green playing ground. It is because the lighting has yellow color content. Such noise is typical “salt-pepper” noise. That is, the number of noisy pixels is small, but they are dispersed around the background rather than concentrated in several regions and their luminance values are relatively high.
- There are many dark green patches around the playing ground board. It is caused by shadow and inconsistent lighting.

In addition, because of other random noise effects in the environment, the image will change slightly every moment. All these facts imply that it is necessary to do color calibration in order to find proper color thresholds.

The camera will introduce nonlinear factor if the field of view (FOV) of the camera is relatively large. The barrel distortion will happen naturally for wide angle lens. According to the specifications of game setting, if assuming the camera is at the height of 2.6 m, the lens view angle is at least 55.57° , it will obviously introduce distortion as shown in Fig. 4-1. To compensate for the kind of distortion in the image, camera parameters are required, which make it necessary to do camera calibration.

Chapter 4 Color and Camera Calibrations

In Section 4.2, we will describe a K-Mean-clustering algorithm for color calibration. Section 4.3 introduces Tsai's model to compute camera parameters.

4.2 K-Mean-clustering Color Calibration

Image segmentation is a low-level image processing task that aims to partition an image into homogeneous regions [DUDA73]. There must be 6 parameters for color-based segmentation. These parameters can be provided by manual selection, but it is better if it can be achieved by an algorithm.

In regards to the color distribution in the *RGB* color space, it is natural to use the clustering algorithm to provide the parameters for color-based segmentation. Clustering is an important unsupervised classification technique where a set of patterns, usually vectors in a multi-dimensional space, are grouped into clusters in such a way that patterns in the same cluster are similar in some sense and patterns in different clusters are dissimilar in the same sense [ANDB73] [HRTG75] [DVJV82] [JAIN88] [TOUT74].

The K-Means clustering algorithm is a widely used clustering algorithm. The following is a brief introduction. The K-Means clustering algorithm was introduced in 1967 by *MacQueen*, and is considered as an unsupervised classification technique [MCQN67]. The algorithm is based on the minimization of the sum of square distances, i.e.

$$\mathbf{J}_j = \sum_{\mathbf{X} \in S_j(k)} \|\mathbf{X} - \mathbf{Z}_j\|^2, \quad j = 1, 2, \dots, K, \quad (4.1)$$

where $S_j(k)$ is the cluster domain for class center $\mathbf{Z}_j$ at the k th iteration.

Mathematical notations are defined below:

Z_j - The center of class j ;

S_j - The cluster domain for class center Z_j ;

N_j - The number of members in the set $S_j(k)$;

J_j - The squared distance from the each member to the cluster center Z_j ;

$\sum J_j$ -The sum of J_j .

A general process of K-Mean clustering algorithm can be divided into three steps as described below.

K-Mean Clustering Algorithm

<1> Select K samples arbitrarily as initial class centers, i.e. $Z_1(k)$, $Z_2(k)$, $Z_3(k)$, ..., $Z_k(k)$; where $k = 1$.

<2> At k th iteration, distribute the pattern sample X among the K classes according to the following rule:

$$X \in S_j(k), \text{ if } \|X - Z_j(k)\| < \|X - Z_i(k)\|, \quad (4.2)$$

for $i = 1, 2, \dots, k, i \neq j$,

Compute

$$Z_j(k+1) = \frac{1}{N_j} \sum_{X \in S_j(k)} X, \quad (4.3)$$

$$J_j = \sum_{X \in S_j(k)} \|X - Z_j(k+1)\|^2, \quad (4.4)$$

Here N_j is the members number of the set $S_j(k)$;

<3> If $Z_j(k+1) = Z_j(k)$, for $j = 1, 2, \dots, k$, stop.

Otherwise go to step 2.

Chapter 4 Color and Camera Calibrations

Before applying the above algorithm in our application, many problems need to be considered and solved in advance.

4.2.1 Initial Class Center Selection

Proper selection of initial cluster centers is important to this algorithm because K-Mean algorithm does not guarantee the global minimum of the objective function [SELM87]. That is, if the initial class center is unsuitable, clustering will be trapped in a local minimum.

It is assumed that there are only eight classes in the image because only eight colors are considered. The RGB value of a pixel near the color blob center is selected for each color as an initial class center. This method seems successful, judging from the experimental results.

4.2.2 Color Sample Collection

In the actual game setting, there is only one orange ball in use, however the ball may move everywhere on the playing field where the lighting may not be uniform. The reflection and shadow of the objects will affect the colors of different parts on the ball. Therefore, one color sample of the ball is not sufficient to yield good results. In order to obtain a more accurate result, many balls are distributed on the playing ground to collect clustering samples. A similar method is applied to analyze the colors of other objects.

4.2.3 Number of Classes in Clustering

This problem is the key to accurate clustering. The K-Mean clustering algorithm can merely decide which class a pixel belongs to, that is, a pixel belongs to one class just because it is nearer to its center than to those of any other's. This may lead to errors in clustering when there is salt-pepper noise, because these noisy pixels will form several new classes and they cannot be detected by observation alone.

Two steps are used to determine how many classes should be used in clustering. First, a raw image is processed using a median filter. This step is expected to remove most of the noises. Second, pixels of the filtered image will be shown in a 3D RGB space. Many 3D blobs can be observed easily. The number of these blobs will determine the number of classes.

4.2.4 Experiment Results and Discussion

Fig. 4-2 Image used for clustering.

The source image Fig. 4-2 is a clipped region within the image of Fig. 4-1.

Chapter 4 Color and Camera Calibrations

4.2.4.1 Result of clustering with eight classes

Fig. 4-3 Image of clustering result with 8 classes

In this experiment, eight classes are used in the clustering since only eight colors can be observed in Fig 4-3. After the clustering algorithm is performed, a pixel RGB value is replaced by the center RGB value of the cluster to which the pixel belongs. Fig. 4-3 shows the result of the cluster algorithm. From the image we can see that the RGB space is segmented into 8 domains.

Table 4-1 Result of K-Mean clustering with 8 classes

Color	RGB Value			Num	J	RGB Std Dev		
White	253.2	255.0	251.4	1677	293468.4	7.8	0.7	10.6
Yellow	227.8	244.2	152.1	435	773834.7	29.6	14.3	26.4
Green	66.0	124.5	50.0	30249	2795888.0	5.2	7.0	3.8
Blue	117.6	162.1	94.8	626	1562974.9	26.1	27.7	32.4
Orange	228.0	144.2	61.3	403	892319.6	31.7	9.9	17.8
Black	63.7	69.2	39.3	946	524759.2	15.7	15.1	9.0
Magenta	253.2	164.4	154.4	169	193179.3	8.5	18.8	26.8
Cyan	163.6	240.6	225.5	175	384059.9	25.3	23.4	31.7

Table 4-1 presents the statistical results of RGB value for each color of interest. We achieved $\sum J = 7420484$.

4.2.4.2 Result of clustering with ten classes

Table 4-2 Result of K-Mean clustering with 10 classes

Color	RGB Value			Num	J	RGB Std Dev		
White	253.9	255.0	252.6	1611	147101.0	5.6	0.8	7.7
Yellow	244.7	252.5	167.2	338	327644.0	16.5	6.5	25.6
Green	65.9	124.5	50.0	30153	2551935.5	4.5	6.9	3.8
Blue	107.9	174.4	131.3	244	538782.1	22.7	27.0	31.0
Orange	235.8	147.4	62.4	357	632428.1	25.4	28.6	17.5
Black	62.1	68.7	39.2	921	407620.7	12.0	15.0	8.6
Magenta	253.2	165.1	153.6	170	200766.2	8.5	19.6	26.9
Cyan	176.9	250.9	241.8	159	177032.4	24.9	10.1	19.8
Shadow1	123.0	138.7	64.9	520	851394.9	24.8	26.0	18.6
Shadow2	180.5	221.6	130.2	207	278280.6	23.5	14.7	24.0

By using the method proposed in Section 4.2.3, ten classes are determined and applied in the clustering. Table 4-2 presents the statistical results of *RGB* value for each color of interest. In the 10 class clustering, class Shadow1 is the pixels around magenta orientation mark and orange ball. Class Shadow2 is the line between the playing field white border and its green surface.

From the Table 4-2, we achieve $\sum J = 6112986$. After the clustering algorithm is performed, a pixel’s RGB value is replaced by the RGB value of the cluster center to which the pixel belongs. Fig. 4-4 shows the results of the 10-class cluster algorithm.

Chapter 4 Color and Camera Calibrations

Fig. 4-4 Image of clustering result with 10 classes

Facts in Fig. 4.4 need to be explained further in order to understand the points described in Section 4.1.

A black patch can be seen in the upper right corner on the playing ground. It is the result of shadow and non-uniform lighting.

There are white spots in the cyan markers. The reason is that the *RGB* values of some pixels within the cyan markers are really close to white, that is, the color of cyan patches on the playing ground is actually classified into two clusters.

The centers of the blue patches seem slightly green. The reason is that: "Blue" is close to "Green" under the yellow lighting condition in our experiment.

Fig. 4-5 Class centers in color clustering with 10 classes

Fig. 4-5 shows the ten class centers marked with '⊗' in color clustering. It is very different from the ideal case of RGB model: the color of lighting adds an offset to each of the ideal colors; shadows and reflections give rise to new color classes.

Comparing the $\sum J$ between 10 classes and 8 classes, we find that the result of color clustering with 10 classes is obviously better than that with 8 classes. Thus defining the number of clusters through differentiating colors by human observation is not an optimal way and can be unreliable.

4.3 Tsai's Camera Calibration Model

There are two reasons why camera calibration is required in the vision system. First, in the vision system, the information of states is calculated in the computer image coordinates, while the information of states in world coordinates has the physical meaning. Hence the information of states must be converted from the computer image coordinates into world coordinates. Second, there is *barrel distortion* in the captured image. Tsai's camera calibration algorithm can remove the distortion of the image [TSAI86] [TSAI87].

4.3.1 Camera Model

Fig. 4-6 Camera geometry with perspective projection

Fig. 4-6 illustrates the Tsai's camera model. The Tsai's camera model differentiates from the linear camera model in its consideration of the distortion. Because the application of this model is a relative standard process, this section only summarizes the procedure of computation in Tsai's model. Relevant mathematical signs are defined as:

Notations in Tsai's Camera Model

(x_w, y_w, z_w) - The 3D coordinates of a point P in the 3D world coordinate system;

(x, y, z) - The 3D coordinates of a point P in the 3D camera coordinate system, which is centered at the point of optical center;

R, T - Rotation and translation transformation matrices from the world coordinate to the camera coordinate;

f - The distance between the image plane and the optical center;

k_1, k_2 - The radial lens distortion parameters;

S_x - The uncertainty image scale factor.

4.3.2 Transformation Steps

The following summarizes the steps of transformation from the world coordinate to the camera coordinate.

Transformation Steps

Step 1 Rigid transformation from the object world coordinate system (x_w, y_w, z_w) to the camera 3D coordinate system (x, y, z) , i.e.

Chapter 4 Color and Camera Calibrations

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \mathbf{R} \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} + \mathbf{T} \quad (4.5)$$

Parameters to be calibrated: $\mathbf{R}, \mathbf{T}$;

Step 2 Transformation from 3D camera coordinates (x, y, z) to ideal (undistorted image coordinates (x_u, y_u)) using perspective projection with pin hole camera geometry

$$x_u = f \frac{x}{z} \quad (4.6)$$

$$y_u = f \frac{y}{z} \quad (4.7)$$

Parameter to be calibrated: effective focal length f .

Step 3 Radial lens distortion: This step introduces nonlinear factor.

$$x_d + d1_x = x_u \quad (4.8)$$

$$y_d + d1_y = y_u \quad (4.9)$$

Here (x_d, y_d) is the distorted or true image coordinate on the image plane, and

$$d1_x = x_d(k_1 r^2 + k_2 r^4), \quad (4.10)$$

$$d1_y = y_d(k_1 r^2 + k_2 r^4), \quad (4.11)$$

Where $r = \sqrt{x_d^2 + y_d^2}$.

Parameters to be calibrated: distortion coefficients k_1 and k_2

Step 4 Real image coordinates (x_d, y_d) to computer image coordinates (x_f, y_f) transformation

$$x_f = s_x d_x^{-1} x_d + c_x, \quad (4.12)$$

$$y_f = d_y^{-1} y_d + c_y, \quad (4.13)$$

where

(x_f, y_f) : row and column # of image pixel in computer frame memory

(c_x, c_y) : computer image coordinates for the origin in the image plane

$$d'_x = d_x N_{cx} / N_{fx}$$

d_x : distance between adjacent sensor elements in the X direction

d_y : distance between adjacent sensor elements in the Y direction

N_{cx} : number of sensor elements in the X direction

N_{fx} : number of pixel in a line as sampled by the computer

Parameters to be calibrated: uncertainty image scale factor s_x and image origin (c_x, c_y) .

The above four steps of transformation in order can be summarized:

$(x_x, y_w, z_w) \rightarrow (x, y, z) \rightarrow (x_u, y_u) \rightarrow (x_d, y_d) \rightarrow (x_f, y_f)$. The parameters to be calibrated are: (1) $\mathbf{R}, \mathbf{T}$, (2) f , (3) k_1, k_2 , (4) $s_x, (c_x, c_y)$. The calibration can be done provided the following parameters are available:

1. d_x, d_y, N_{cx}, N_{fx} . These parameters are provided by the camera specification.
2. Many position pairs. A position pairs consists of a point's world coordinates (x_x, y_w, z_w) and its corresponding image coordinates (x_f, y_f) .

In real world tracking, the position calculation of the objects in the world coordinates is achieved in the reverse order. First the image processing provides the position of the objects in pixel coordinates (x_f, y_f) , then the positions of the objects are obtained using camera calibration parameters.

4.3.3 Acquisition of Matching Points

The Tsai's approach requires usually about 50 sample points for computation. Their world coordinates and the coordinates of their matching points in the image plane should be known first. It is generally a time-consuming task. To complete this work autonomously, a grid pattern as shown in Fig. 4-7 is designed and applied to finding the coordinates of the sample points in the image by blob analysis.

Fig. 4-7 The image of the calibration pattern

The pattern consists of 6 rows and 8 columns of evenly placed squares, with each square having the dimension of 10x10 cm. The distance between 2 neighboring squares in the x -direction is 22 cm, and 15cm in the y -direction.

The positions of the points in the world coordinates can be easily obtained. The positions of the points in the image plane depend on the selection of image processing techniques. At first, we intend to do the matching work by *corner detection*. However, the precision obtainable using corner detection is 1 pixel, which is fairly high, and the corner results

are not very satisfactory due to the shape of the blobs in the captured image. There is some noise.

Compared with *corner detection*, *blob analysis* has no such shortcomings. The 48 centers of the squares in the pattern are selected as the samples of the camera calibration. The 8-connected-neighbour is used in the blob analysis, and parameters such as compactness and area are used to filter out some unwanted results. The result of the blob analysis contains the centers of gravity of the blobs in the image. Hence, we get the image coordinates.

Then acquisition of matching points can be divided into three steps:

1. Calculation of the centers of squares in the world coordinates;
2. Calculation of the blob centers in the image by blob analysis;
3. Matching the squares' centers with the blob centers.

The world coordinates of the centers of the squares can be easily obtained because of the pattern design. Bilinear interpolation is used to compute all the other centers of the squares. Only the world coordinates of the four corner points are measured with $z_w = 0$.

Blob analysis can be used by the method described in the last chapter.

The remaining work is to match the world coordinates with the image coordinates, i.e. to match the results of the blob analysis with the results of the bilinear interpolation. It is not so easy to directly match the points. We found an indirect way. Since the squares in the

Chapter 4 Color and Camera Calibrations

patterns cloth we designed are sparsely displaced, one can imagine that the distance between the centers of the gravity is fairly large. On the basis of this feature, we do the matching in two steps as described below:

We first roughly measure the coordinates of the centers of the four corner squares in the image, then get the remaining center's coordinates by bilinear interpolation. The following pseudo code describes the estimation of the rough blobs centers.

Estimation of Rough Blobs Centers

```
// The following denotes the image of the calibration pattern
// o(P1)-----o(P2)
// | . . . . . |
// | . . . . . |
// | . . . . . |
// o(P3)-----o(P4)
// P1, p2, p3, p4 are the blob centers of the four corner blobs,
// the positions of these corners are measured.
PATTERN_X_NUM: the number of pattern in the x direction
PATTERN_Y_NUM: the number of pattern in the y direction

delta_x = (float)1/((float)PATTERN_X_NUM-1);
delta_y = (float)1/((float)PATTERN_Y_NUM-1);
for(i = 0; i<PATTERN_X_NUM; i++)
    for(j = 0; j<PATTERN_Y_NUM; j++){
        x = i*delta_x; y = j*delta_y;
        RoughBlobsCenters[i][j].x = (1-x)*(1-y)*P[0].x
                                   + ( x)*(1-y)*P[1].x
                                   + (1-x)*( y)*P[2].x
                                   + ( x)*( y)*P[3].x;
        RoughBlobsCenters[i][j].y = (1-x)*(1-y)*P[0].y
                                   + ( x)*(1-y)*P[1].y
                                   + (1-x)*( y)*P[2].y
                                   + ( x)*( y)*P[3].y;
    }
```

We then match the rough results with the precise result by the blob analysis using distance as the criterion. Fig. 4-8 illustrates the matching algorithm.

Fig. 4-8 Illustration of the matching algorithm

Now the rough centers and the precise gravities of the blobs are matched. Because the order of the rough results is just the same as the one of the world coordinates, the sorted pairs of the world coordinates and the image coordinates are realized.

4.3.4 Result and Evaluation

We achieved matching data from the blob analysis and the matching algorithm. For details, **cc_cd.dat** ($z_w = 0$) in Appendix D shows the original data of a DXC-9000

Chapter 4 Color and Camera Calibrations

camera. From these data, Tsai’s algorithm can be used to obtain the camera’s internal and external parameters. Table 4-3 shows the calibration result of SONY DXC-9000. All these data are precise except f and T_z . We find that f from the camera calibration is outside the specification of the camera. We can also see that T_z is about 5.4 meters, actually it is in the range of 2.5~3 meters, while T_z/f is very close to our estimation. In a real application, T_z/f is what we really want.

Table 4-3 Camera parameters of SONY DXC-9000

DXC-9000 Camera Parameters											
Coplanar calibration (full optimization)											
camera type: SONY DXC-9000											
data file: D:\soccer\cc_cd.dat (48 points)											
$f = 11.626337$ [mm]											
$kappa1 = 3.944832e-003$ [1/mm^2]											
$T_x = -1396.269977, \quad T_y = -943.058893, \quad T_z = 5368.415709$ [mm]											
$R_x = 3.328443, \quad R_y = 3.144857, \quad R_z = 0.040548$ [deg]											
R											
<table><tr><td>0.998494</td><td>0.002479</td><td>0.054809</td></tr><tr><td>0.000707</td><td>0.998315</td><td>-0.058021</td></tr><tr><td>-0.054861</td><td>0.057972</td><td>0.996810</td></tr></table>			0.998494	0.002479	0.054809	0.000707	0.998315	-0.058021	-0.054861	0.057972	0.996810
0.998494	0.002479	0.054809									
0.000707	0.998315	-0.058021									
-0.054861	0.057972	0.996810									
$s_x = 1.000000$											
$C_x = 161.941712, \quad C_y = 130.657236$ [pixels]											
$T_z / f = 461.746113$											

How to evaluate if the calibration works? This is done by examining the error of the object’s real position measured in the field and its estimation from the tracking system. It is found that the error is less than 1 cm, equivalently to approximately 1 pixel.

Tsai’s algorithm also provides four kinds of evaluation [TSAI86] [TSAI87]. Table 4-4 contains the evaluation data using the full optimization of Tsai’ Camera Calibration of a DXC-9000 camera.

Table 4-4 The evaluation data by the full optimization of DXC-9000

	Mean	StdDev	Max	SSE
distorted image plane error	0.3046 [pix]	0.1752 [pix]	0.7078 [pix]	5.8976 [pix^2]
undistorted image plane error	0.3136 [pix]	0.1835 [pix]	0.7549 [pix]	6.8056 [pix^2]
object space error	3.1984[mm]	1.7277 [mm]	6.6156 [mm]	509.7056 [mm^2]
Normalized calibration error	0.768350			

Table 4-5 The evaluation data by the full optimization of JAI 70 AC

	Mean	StdDev	Max	SSE
distorted image plane error	0.3173[pix]	0.1721[pix]	0.7318 [pix]	6.2255 [pix^2]
undistorted image plane error	0.3321[pix]	0.1792[pix]	0.7549 [pix]	6.8056 [pix^2]
object space error	3.1984[mm]	1.7277[mm]	7.2913 [mm]	631.3282 [mm^2]
Normalized calibration error	0.813582			

Table 4-5 contains the evaluation data by the full optimization of Tsai’ Camera Calibration of a JAI 70 AC camera.

These data prove that the error in the real world is less than 1 cm. By using the parameters determined, we obtain steady state data of the robots and the ball.

CHAPTER 5

Introduction to Path Planning

In this and chapter 6, we discuss the path planning problem in autonomous robots. This chapter introduces several classical algorithms for the path planning, including *Distance Transform*, *Lee's algorithm*, *Artificial Potential Fields*, and *Roadmap Method*. We also introduce path planning problem arising from the robotic soccer field.

5.1 Introduction

One of the ultimate goals in Robotics is to create autonomous robots. Such robots will accept high-level descriptions of tasks and will execute them without further human intervention. The input descriptions will specify *what* the user want done rather than *how* to do it [LTMB91]. Obviously, the goal of robotics is consistent with the goal of the RoboSoccer.

There are a wide variety of application domains for autonomous robots, including manufacturing, construction, waste management, space exploration, traffic planning, undersea work, medical surgery and RoboSoccer [PARK99].

The path planning problem deals with robot motion planning, which decides for the robot what motions to perform in order to achieve goal arrangements of physical objects. In other words, path planning is concerned with planning a feasible path for a robot in the environment. The problem can be defined thus, given an initial position and orientation and a goal position and orientation, generate a path specifying a continuous sequence of positions and orientations avoiding contact with the obstacles, starting at the initial position and orientation, and terminating at the goal position and orientation. Report failure if no such path exists [LTMB91].

Static path planning is concerned with path planning in a static environment. Dynamic motion planning is concerned with motion planning in a dynamic environment. This thesis focuses on the simpler static path planning and we do not attempt to solve the problem of dynamic motion planning in RoboSoccer. However, a little comparison of these two issues will clarify the problem further.

5.1.1 Dynamic Planning and Static Planning

There are two kinds of planning, dynamic motion planning and static path planning. Static path planning only has to provide a feasible path. Dynamic motion planning has to consider the time factor in addition to finding the path. The dynamic motion planning problem is much more difficult than the static path planning problem. Actually, after the tracking module captures the state information in the field, the robot soccer problem is a classic dynamic motion planning problem because the intelligent module sends intentions to the planning module such as kicking the ball in a certain direction, intercepting the ball

Chapter 5 Introduction to Path Planning

from the opponent robot, etc. Then the planning module plans a path with temporal information. After that, the control module sends commands to the corresponding robots. The plan may not be accomplished in the competition because the environment could change, for example, an opponent robot may be in the way of the attacking robot.

Fiorini and *Shiller* utilize the concept of velocity obstacle (VO) to solve the dynamic motion planning. The method is demonstrated for point and disk robots among static and moving obstacles, and for an automated vehicle in an Intelligent Vehicle Highway System scenario. The trajectory from start to goal is computed by searching a tree of feasible avoidance maneuvers at discrete time intervals [FORN88] [FORN95].

The above method is a near-optimal algorithm for dynamic environments. But it is not feasible for the RoboSoccer problem. In the RoboSoccer domain, because the environment is very complex, it is very hard to find a near-optimal algorithm for dynamic motion planning. Researchers like *Peter Stone* have presented other ways such as layered learning to win the game. They plan a practical real-time robot motion control algorithm without any consideration for the complexity of the RoboSoccer environment [STON98] [STON00]. It is not optimal and it is difficult to evaluate the algorithm.

What are the difficulties in the dynamic motion planning for RoboSoccer? Static path planning is a kinematic problem, involving the computation of a collision-free path from start to goal, whereas velocity planning is inherently a dynamic problem, requiring the consideration of robot dynamics and actuator constraints. It is shown that dynamic

motion planning for a point in the plane, with bounded velocity and arbitrary number of obstacles, is intractable and NP-hard [CANN87]. In addition to the computational difficulty, this problem is not guaranteed to have a solution due to the uncertain nature of the environment, since a solution computed at time t_0 may be infeasible at a later time [SNBN88]. And what is more, long term optimal dynamic motion planning is nearly impossible, considering the behaviors of the higher intelligent modules of both competing teams.

Nowadays, the intelligent control module of most teams pays more attention to learning algorithms, and less attention to the dynamic motion planning. But we are sure that it will lead to a great progress in RoboSoccer if the dynamic motion planning problem is solved. In this thesis, we start research on the static path planning problem as a first step in that direction. The following sections will introduce the required background for static path planning.

5.1.2 The Basic Problem

The following is the basic motion planning problem.

Let $\mathbf{A}$ be a single rigid object, the robot, moving in a Euclidean space $\mathbf{W}$, called *workspace*, is represented as $\mathbf{R}^2$. Let $\mathbf{B}_1, \dots, \mathbf{B}_q$ be fixed rigid objects distributed in $\mathbf{W}$. The $\mathbf{B}_i$'s are called obstacles. Assume that both the geometry of $\mathbf{A}$, $\mathbf{B}_1, \dots, \mathbf{B}_q$ and the locations of the $\mathbf{B}_i$'s in $\mathbf{W}$ are accurately known. Assume further that no kinematic constraints limit the motions of $\mathbf{A}$ (we say that $\mathbf{A}$ is a *free-flying object*). The problem

Chapter 5 Introduction to Path Planning

of static motion planning can be stated as: Given an initial position and orientation and a goal position and orientation of $\mathbf{A}$ in $\mathbf{W}$, generate a path τ specifying a continuous sequence of positions and orientations of $\mathbf{A}$ avoiding contact with the $\mathbf{B}_i$'s, starting at the initial position and orientation, and terminating at the goal position and orientation [LTMB91].

5.1.3 Kinematic Constraints and Robot Model

Actually not all of the robots are free-flying robots, which can translate and rotate freely in the workspace. There are kinematic constraints, which can be classified into two types: holonomic constraints and nonholonomic constraints [LMND86] [LMND87].

Holonomic constraints only affect a subset of configurations and do not fundamentally change the path planning problem, while nonholonomic constraints in the path planning problem can greatly affect the motion of a car-like mobile robot [BRQN89]. How to tell the difference between these two constraints? If the constraint is integrable, an equation involving velocity parameters will reduce it to a holonomic constraint. The following are the definitions [LTMB91]:

1. A scalar constraint of the form $F(\mathbf{q}, t) = 0$ ($F(\mathbf{q}, t) < 0$ or $F(\mathbf{q}, t) \leq 0$), where F is a smooth function with non-zero derivative, is called a **holonomic** equality constraint (**holonomic** inequality constraint).
2. A non-integrable scalar constraint of the form: $G(\mathbf{q}, \dot{\mathbf{q}}, t) = 0$ ($G(\mathbf{q}, \dot{\mathbf{q}}, t) < 0$ or $G(\mathbf{q}, \dot{\mathbf{q}}, t) \leq 0$), where G is a smooth function, is called **nonholonomic** equality constraint (**nonholonomic** inequality constraint).

A common nonholonomic constraints model is shown in Fig. 5-1, and this model is also used in our problem. Fig 5-1 (a) shows a generic car-like robot, and Fig 5-1 (b) shows a particular unicycle vehicle. Fig 5-1 (c) shows a left turn or counterclockwise circle and a right turn or clockwise circle, both with the minimum turning radius for a robot.

Fig. 5-1 The model of the robot and its constraints.

Consider the car-like robot, Let ϕ be the angle between the main axis of the car and the velocity vector of the midpoint between the two front wheels. ϕ is the steering angle. In general, mechanical stops in the steering gear constrain ϕ in such a way that

$|\phi| \leq \phi_{\max} < \frac{\pi}{2}$, which is called *limited steering angle*. The midpoint between the two

rear wheels must follow a curve whose curvature is upper-bounded by: $\frac{1}{\rho_{\min}} = \frac{1}{L} \tan \phi_{\max}$,

which is called *minimal turning radius* of the car, where L is the distance between the

Chapter 5 Introduction to Path Planning

front wheel and the rear wheel. Fig. 5-1 (c) shows a robot model with bounded turning radius, which is also the main constraint in our problem.

5.1.4 C-Obstacles

The concept of C-obstacles is widely used and adopted in many motion planning algorithm [LTMB91]. Every obstacle $B_i, i=1$ to q , in the workspace W maps in C to a region: $CB_i = \{q \in C / A(q) \cap B_i \neq \emptyset\}$, which is called a C-obstacle region. The union of

all the C-obstacles: $\bigcup_{i=1}^q CB_i$ is called the C-obstacle region, and the set

$C_{free} = C \setminus \bigcup_{i=1}^q CB_i = \{q \in C / A(q) \cap (\bigcup_{i=1}^q B_i) \neq \emptyset\}$ is called the free space. Any

configuration in C_{free} is called a free configuration. Using the concept of C-obstacle, the robot can be reduced into a point and the obstacles become larger.

Fig. 5-2 C-obstacle

Fig. 5-2 illustrates the concept of C-obstacle. The robot $\mathbf{A}$ is a free-flying robot and the obstacles $\mathbf{B}_1$ and $\mathbf{B}_2$ are rectangles. The C-obstacle $\mathbf{CB}_i$ is obtained by “growing” the corresponding workspace obstacle $\mathbf{B}_i$ (a rectangle) by the shape of $\mathbf{A}$. Planning a motion of $\mathbf{A}$ relative to $\mathbf{B}_i$ is equivalent to planning a motion of a point relative to $\mathbf{CB}_i$.

5.2 Classical Algorithms

This section introduces several classical algorithms commonly used in path planning.

There are several assumptions in these algorithms:

- The robot is the only moving object in the workspace.
- The kinematic properties of the robot are ignored.
- The robot is a single rigid object, i.e. an object whose points are fixed with respect to each other.

5.2.1 Grid-based Algorithms

Here we assume the obstacles are polyhedral, and robot is a point. Lee’s algorithm is widely used in mobile robot for path planning. It provides a nearly optimal answer to the path planning problem by dividing the 2D space into grids. The following example of Lee’s Algorithm is about Maze Routing [CYLE61]. Maze routing is the problem of finding the least expensive path between a source, $\mathbf{S}$, and a target, $\mathbf{T}$, with the presence of obstacles. Obstacles could also include previously routed wires. The path planning procedure of this algorithm is like wave propagation, which means that the points on the same wave of the same distance to the source $\mathbf{S}$. Firstly, the wave is generated around the

Chapter 5 Introduction to Path Planning

source S , then a next step wave is iteratively generated from the last step wave until the target T is met. With a path length/cost of n , Lee's algorithm has $O(n^2)$ complexity.

Fig. 5-3 shows the algorithm. Fig. 5-3 (a) shows the initial condition. Fig. 5-3 (b) shows the procedure of the algorithm like wave propagation until the target is met.

Fig. 5-3 Find a shortest route between A and B

Another commonly used algorithm is Distance Transform, which is similar to Lee's algorithm. Both of them search for an optimal path in grid-based maps. The principle of both of these algorithms is wave front propagation [JRVS83] [JRVS88].

5.2.2 Potential Fields Method

This approach allows sensory measurements of obstacles to be used as feedback during the execution of paths planned with another planning algorithm. It is very useful in real-time motion control. Compared with the grid-based methods, the potential field method is totally different. The ideas come from *Classical Mechanics*. It treats a robot as a point in

configuration space as a particle under the influence of an *artificial potential field* U whose local variations are expected to reflect the “structure” of the free space. The potential function is typically defined over free space as the sum of an *attractive* potential pulling the robot towards the goal configuration and a *repulsive* potential pushing the robot away from the obstacles. Motion planning is performed in an iterative fashion. At each iteration, the artificial force induced by the potential function at the current configuration is regarded as the most promising direction of motion, and path generation proceeds along this direction by some increment [KHTB86]. Potential field methods are often referred to as “local methods” because of local minima during searching.

5.2.3 Roadmap Methods

The roadmap methods are different from the grid-based methods and the potential field methods. Its approach to path planning consists of capturing the connectivity of the robot's free space in a network of one-dimensional curves, called the roadmap, lying in the free space $\mathbf{C}_{free}$. The Roadmap methods include Voronoi Graph [ODNL83] [ODNL84], Visibility Graph [NSSN69], and Randomized Roadmap [AMTO96]. We only introduce Voronoi Graph for it simplifies the problem into searching a graph, which is a very inspiring idea.

The basic visibility graph algorithm [LTMB91]

The visibility graph method usually has three steps:

1. Construct the visibility Graph $\mathbf{G}$.
2. Search $\mathbf{G}$ for a path from $\mathbf{q}_{init}$ and $\mathbf{q}_{goal}$ (initial and goal configurations).

Chapter 5 Introduction to Path Planning

3. If path is found, return it; otherwise, indicate failure Construction of the Visibility Graph (a naive algorithm).

Fig. 5-4 A visibility graph G

Fig 5-4 shows the visibility graph G of a simple configuration space in which CB consists of two disconnected regions, which are the shaded areas. The links of G also include the edges of CB . The bold line is the shortest path belonging to the close set of C_{free} between q_{init} and q_{goal} .

The vertices of the obstacles are considered as nodes. The nodes are linked with straight line segments if they do not intersect the interior of the C -obstacle region. The initial and the goal configurations q_{init} and q_{goal} are considered as nodes and are linked with straight line segments to other nodes, if the line exists in C_{free} . The algorithm given above, requires time $O(n^3)$ [LZPR79] where n is the total number of vertices of CB . There are

more efficient algorithms performing in time $O(n^2 \log n)$ [LEED78], $O(n^2)$ [WLZL85] and $O(k + n \log n)$ [GHSH87], where k is number of edges in G .

Maybe searching the visibility graph G is the most important step in the path planning algorithm, and it can be done using various graph searching techniques. Whenever a path exists, it is returned; otherwise, a failure is reported. The search is done in time $O(n^2)$ or $O(k \log n)$ depending on the search technique. The famous algorithms like A* algorithm (A special format is *Dijkstra's* routing algorithm) can be used for searching [NSSN80].

5.3 Model of Static Path Planning in RoboSoccer

Fig. 5-5 A static model of the RoboSoccer field.

Chapter 5 Introduction to Path Planning

Fig 5-5 shows the model of our problem, a static path planning model in RoboSoccer. The field size is 274 by 152.5 meters, exactly as the field defined by the rule of the RoboCup small robot league. In front of each goal, there is a penalty point. There are n (here n is 10) robots, which are robot No.1 to No.9 and robot A at the initial position $\mathbf{q}_{init}$, and one ball at the goal position $\mathbf{q}_{goal}$ in the field. The robot R 's initial state is $\mathbf{q}_{init}(x_0, y_0, \theta_0)$, and the target ball's state is $\mathbf{q}_{goal}(x_f, y_f, \theta_f)$. The robot's radius is r , and the minimum turning radius is ρ_{min} . After C-obstacle transformation, the obstacle's radius is $R = 2r$.

The Problem Description

Assumption:

1. Each robot's is a circle with radius r_i .
2. The robot's minimum turning radius is ρ_{min} .
3. Only one robot R can move, the other robots are static obstacles.
4. The robot R 's initial state is $\mathbf{q}_{init}(x_0, y_0, \theta_0)$, and the goal state is $\mathbf{q}_{goal}(x_f, y_f, \theta_f)$.

Requirement: To find a shortest path in the static environment, from initial state $\mathbf{q}_{init}$ to the goal state $\mathbf{q}_{goal}$ with obstacle avoidance.

Let's analyse the problem description.

1. If there is no assumption 2, and assumption 1 is modified such that the robot is a square, the problem can be solved by a visibility graph.

2. If the assumption 3 is modified such that all robots can move, the problem is dynamic motion planning problem, which is a very difficult problem.

So the first step is to consider C-obstacle, so the obstacle radius is now $2r$. But we have to consider the minimum turning radius $\rho_{\min}$.

1. If $\rho_{\min} \leq 2r$, then the obstacle's radius is $2r$, and the moving robot can turn around the obstacle.
2. If $\rho_{\min} > 2r$, we choose $\rho_{\min}$ as the obstacle's radius based on *Dubin's theorem*, which will be introduced in the next chapter. This is an approximate modification to the assumption of the problem. This approximation will make it easy to deal with the problem.
3. If $\rho_{\min} \gg 2r$, this will be a hard problem, which may have no solution. Fortunately in real applications, this will not happen.

Thus the problem is equivalent to finding the shortest path for a robot to move from an original position and orientation to the final position and orientation, with circle obstacles avoidance. This problem is similar to the visibility graph except that we have to consider the kinematic constraints and that the obstacles are circles. Like the visibility graph algorithm, searching a path is also the most important in this problem.

CHAPTER 6

A Path Planning Algorithm

In this chapter, a novel path planning algorithm is presented based on genetic algorithm. First, we will introduce the Dubin's algorithm [DUBN57], which provides the shortest path if there are no obstacles. Then we introduce the main idea of our algorithm, which is called the "rope solution" because the path is like a rope in an environment with obstacles. Note that it is not promised to provide the optimal path. Since, the main idea of the algorithm is about searching, a genetic algorithm is developed to solve this problem.

6.1 Dubin's Theorem: Shortest Path without Obstacles

The model of a car-like robot considered here describes the two principal kinematic constraints of a usual car.

1. The first one is the rolling without slipping constraint which obliges the vehicle to move tangentially to its main axis.
2. The second constraint is due to the bound on the turning wheels' angle and it prevents the car from moving on trajectories whose radius of curvature is lower than a given threshold.

A configuration of the car is represented by a triple $(x, y, \theta) \in \mathbf{R}^2 \times S^1$ as shown in Fig. 6-1, where (x, y) are the coordinates of a reference point of the robot with respect to a Cartesian frame, and θ is the angle between the positive x-axis and the robot's main axis.

Fig. 6-1 The Dubin's car model

Therefore, the kinematics of the vehicle is described by the differential system involving two control parameters u_1 and u_2 which represent respectively the algebraic value of the linear and angular velocities [LMND98].

The kinematics of the Dubin's car is described by [DUBN57]:

Chapter 6 A Path Planning Algorithm

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \cos \theta \\ \sin \theta \\ 0 \end{pmatrix} u_1 + \begin{pmatrix} 0 \\ 0 \\ 1/R \end{pmatrix} u_2 \quad (6-1)$$

Dubin's theorem provides the shortest path without obstacles for a particle moving forward in the plane with a constant linear velocity ($u_1 = k$). In this case, as the modulus of the linear velocity is kept constant, the shortest path problem is equivalent to the minimum-time problem [Dubins57].

Fig. 6-2 Two types of optimal paths in Dubin's theorem

Dubin proved the existence of shortest paths of the minimum-time problem. He showed that the optimal trajectories are necessarily made up of arcs of circles C of minimal turning radius and line segments S . Therefore, he proved that any optimal path must be of one of the following two path types:

$$\{C_a C_b C_c, C_a S_d C_e\}, \text{ where: } 0 \leq a, e < 2\pi, \pi < b < 2\pi, \text{ and } d \geq 0, \quad (6-2)$$

where C_a is an arc of the initial circle, C_c is an arc of the target circle, C_b is an arc of the circle which is tangential to both the initial circle and the target circle, S_d is a line segment which is tangential to both the initial circle and the target circle.

Fig. 6-2 shows the optimal path in Dubin's problem. In Fig. 6-2 (a) the optimal path is $\widehat{AB} + \widehat{BC} + \widehat{CF}$, which is $C_a C_b C_c$ type. A is the initial point, F is the final point, B, C, D, and E are the tangent points between 2 tangent circles. In Fig. 6-2 (b) the optimal path is $C_a S_d C_e$ type.

6.2 "Rope Solution": Shortest Path with Obstacles

Dubin's algorithm provides the optimal path without obstacles and it is a concatenation of arcs and line segments, but what if there are obstacles as Fig.5-5 shows?

We have introduced our model of Fig.5-5 in chapter 5. There are 9 obstacles. So if you directly apply Dubin's algorithm, the path will intercept with the obstacles. A simple idea is to form a path with a concatenation of arcs and line segments, while the constraints are still satisfied at the same time.

Let's first draw a random path from $\mathbf{q}_{init}$ to $\mathbf{q}_{goal}$ like in Fig 6-3 (a). Every point of the rope is in the free space $\mathbf{C}_{free}$. This path could not be a feasible path because you can see that it cannot satisfy the nonholonomic constraints and the initial orientation and final orientation may not be the same as the requirements in the problem. One can think of the

Chapter 6 A Path Planning Algorithm

path as a “loose” rope with its initial and final points fixed on $\mathbf{q}_{init}$ and $\mathbf{q}_{goal}$. And after the following transformation, it would be a feasible path.

Fig. 6-3 (a) A random path like a rope. (b) A feasible path is like a tight rope.

If you fix the one end of the rope in $\mathbf{q}_{init}$ and another end of the rope passes through $\mathbf{q}_{goal}$, and you draw one end of the rope $\mathbf{q}_{goal}$, then the rope become tighter and tighter until it cannot be drawn further. There could only be one such rope for the initial conditions. This method provides a feasible path for the problem, but could we find the shortest path?

Or equivalently, “is the shortest path among the paths provided in this method?” The answer is not guaranteed, the shortest path could be provided in this way with one exception. If we observe carefully, we would found that the path we provides is $C_a S_b C_c S_d \dots S_s C_t$, which is similar to the type $C_a S_d C_e$ in Fig.6-2 (b). But from Dubin’s theorem, we know that we miss the consideration of the type of $C_a C_b C_c$. Maybe the

shortest path has the form like $X_a X_b X_c X_d \dots X_s X_t$, here X could be an arc C or a line segment S .

Fig. 6-4 The optimal path $\widehat{AB} + \widehat{BC} + \widehat{CF}$

Fig. 6-4 shows the same example as in Fig 6-2 (a), but focuses on the comparison with two kinds of path in Dubin's problem. All the five circles above are auxiliary and virtual circles, each with the same radius R , if the distance $O_1 O_2$ is less than $4R$, there could exist the path $C_a C_b C_c$, with C_b being an arc of a virtual circle. The optimal path is $\widehat{AB} + \widehat{BC} + \widehat{CF}$, which is of $C_a C_b C_c$ type; another path is $\widehat{AD} + DE + \widehat{EF}$, which is of $C_a S_d C_e$ type.

Chapter 6 A Path Planning Algorithm

So from the above analysis, we know that we should draw some “virtual” circles of the same radius of minimal turning radius $\rho_{\min}$ in the initial model plane with the “real” circles of obstacles. Using the same “rope solution” as in the initial model, we can find the shortest path in the modified model plane. But these circles would intercept each other, and the property of the “real” circle and “virtual” circle are not the same. The modified model is quite complex.

So we would like to divide our solution into 2 steps:

1. The first step is to find the shortest path among the form of $C_a S_b C_c S_d \dots S_s C_t$;
2. The second step is to find the shortest path among the form of $X_a X_b X_c X_d \dots X_s X_t$, which is also the shortest path.

In this thesis, we focus on the first step. The second step will be presented as an area for future research. From now on, when the shortest path is mentioned, its form is $C_a S_b C_c S_d \dots S_s C_t$.

6.3 Description of a Path

If we consider the initial random path in Fig.6-3 (a), we would find that (i) the number of such kind of random paths is countless, and (ii) there is only one “tight” path for the initial condition. For Fig.6-2, we can describe the paths in plain words:

- 1) Turn *counterclockwise* around one of the initial circle;
- 2) Drive to the *left* side of Obstacle 1, and turn *clockwise* around Obstacle 1;
- 3) Drive to the *left* side of Obstacle 5, and turn *clockwise* around Obstacle 5;

- 4) Drive to the *right* side of Obstacle 7, and turn *counterclockwise* around Obstacle 5;
- 5) Drive toward the final target, and turn *clockwise* around one of the target circle.

All the paths in Fig.6-3 can be described in these words, so the “structures” of the paths in Fig. 6-3 are the same.

If we describe the path like the above, there exists only one shortest path of the form of $C_a S_b C_c S_d \dots S_s C_t$.

It is very interesting and helpful to find such feature of the problem. From the observation, the path can be described by such words as “drive left”, “turn clockwise around the obstacle”, etc., so the problem becomes one of searching for the shortest feasible path.

6.4 Searching and Genetic Algorithm

Now we know that the searching is almost the most important step in finding the path. If we treat the obstacle as points, and do not consider the kinematic constraints, then the problem can be solved by the roadmap method. And one of the most important steps in visibility method is to build a Graph G . After the graph G is found, one can use the searching algorithm to find the shortest path. Is there such kind of G in our problem? The answer is yes, but it is very complicated because the points in the visibility method become circles. And each circle can have several points on it.

Chapter 6 A Path Planning Algorithm

6.4.1 Relationships between a Path and a Circle

Fig.6-5 (a) Two ways around a circle, (b) The relationships between a path and a Circle.

There are only two possible ways to travel around a circle, in clockwise (+1) or counter-clockwise (-1) direction. Fig. 6-5 (a) shows these two ways.

In Fig. 6-4 (b), if we consider the relationship between a path and a circle obstacle, there are four possible relations:

1. The feasible path is tangential to the circle, and the moving direction in the circle is counterclockwise;
2. The feasible path is tangential to the circle, and the moving direction is clockwise;
3. The path is not tangential to the circle, which means that the robot do not move around the circle;
4. The path intercepts the circle. This kind of path is not feasible.

So, there are only three kinds of relationships between a path and an obstacle.

6.4.2 Complexity of Problem

Let's consider the complexity of the problem with n obstacles circles. As we know, in the initial position, there could be two possible directions, left or right, so are the directions in the final position. And from above analysis, there could be 3 selections when the robot encounters an obstacle. Considering n obstacles, the enumeration is $n!$.

So the complexity of the problem is

$$2 \times 2 \times 3^n \times n! = 4 \times 3^n n! \quad (6-3)$$

In our problem, n is equal to 9. So there are $N = 4 \times 3^9 \times 9! = 28,570,268,160$ possible path. Of course, there could be a lot of infeasible paths among them. From the above analysis, we know the problem is nonpolynomial.

6.4.3 Traveling Salesman Problem

The searching is similar to the traveling salesman problem or TSP [TSPH02]. The TSP is as follows: given a finite number of "cities" along with the cost of travel between each pair of them, find the cheapest way of visiting all the cities and returning to your starting point. The travel costs are symmetric in the sense that traveling from city X to city Y costs just as much as traveling from Y to X ; the "way of visiting all the cities" is simply the order in which the cities are visited. To put it differently, the data consist of integer weights assigned to the edges of a finite complete graph; the objective is to find a Hamiltonian cycle (that is, a cycle passing through all the vertices) of the minimum total weight. In this context, Hamiltonian cycles are commonly called *tours*. A lot of mathematicians study this problem. Eventually, the TSP gained popularity as the

Chapter 6 A Path Planning Algorithm

prototype of a hard problem in combinatorial optimization: examining the tours one by one is out of the question because of their large number, and no other idea was on the horizon for a long time.

This problem cannot be solved by traditional methods that rely on knowing the properties of the path segments. For example, a tour of 9 cities TSP 3 – 2 – 5 – 7 – 6 – 1 – 9 – 8 – 4 would be represented by the string 3 2 5 7 1 9 8 4. For n -town symmetric problem (the distance from town i to town j is the same as the distance from town j to town i), there are $(n-1)!/2$ unique tours which become large very quickly making the solution of this problem impractical using exhaustive search.

Genetic algorithm is a good solution for the TSP. So it is natural to try to solve the path planning problem by genetic algorithm [MSCT89] [WHTL92] [DZBR94].

6.4.4 Genetic Algorithm and TSP

The basic principles of genetic algorithm were first proposed by Holland [HLLD75]. Genetic algorithm or GA is inspired by the mechanism of natural selection where stronger individuals are likely the winners in a competing environment. Here, GA uses a direct analogy of such natural evolution. Through the genetic evolution method, an optimal solution can be found and represented by the final winner of the genetic game. Like the evolution in the natural world, the most important operators in GA are *Crossover* (Recombination), *Mutation*, and *Selection* according to the fitness value. Throughout a genetic evolution, the fitter chromosome has a tendency to yield good

quality offspring which means a better solution to any problem [GLDB85]. In the genetic algorithm, we emulate this feature. The following is the conventional GA:

1. $t = 1$
2. initialize population $P(t)$
3. $t = t+1$
4. if termination criterion is achieved go to step 9
5. select $P(t)$ from $P(t-1)$
6. crossover upper module of $P(t)$
7. mutate upper module of $P(t)$
8. goto 3
9. output best and stop

For the TSP, the crossover operators were *partially matched crossover* (PMX), *order crossover* (OX) and *cycle crossover* (CX). The *inversion mutation* is used as mutation operation [DVIS85] [GLDB85] [SMTH85]. The final stage of the process is the replacement strategy and we shall consider both the elitist strategy, in which the best chromosome from the parent population replaces the worst offspring, and $(\mu + \lambda)$ selection, with $\mu = \lambda$, that is, the best of the offspring and parents are chosen to form the next generation [GLDB85].

6.4.5 Hierarchical Genetic Algorithm

In section 6.3, we described a path. It is not easy to describe it like a TSP string. But we can describe the path like

Chapter 6 A Path Planning Algorithm

$$\{0(-1)1(+1)5(+1)7(-1)10(+1)\} \text{ or } \begin{Bmatrix} 0 & 1 & 5 & 7 & 10 \\ -1 & +1 & +1 & -1 & +1 \end{Bmatrix} \quad (6-4)$$

Here “0” is the initial position, “10” is the target position, “1”, “5”, “7” denote the obstacle numbers. “-1” means counterclockwise direction, and “+1” means clockwise direction. So we can see that the upper part of the string is like a *tour* of TSP, and the lower part of the string is about direction. So you can see the hierarchy from this kind of representation. This is called hierarchical genetic algorithm (HGA). You can also find analogy of HGA in the natural world. The basic steps for HGA in this chapter is as follows:

1. $t = 1$
2. initialize population $P(t)$
3. compute fitness $P(t)$ with penalty strategy
4. $t = t+1$
5. if termination criterion is achieved go to step 12
6. select $P(t)$ from $P(t-1)$
7. crossover upper module of $P(t)$
8. mutate upper module of $P(t)$
9. crossover lower module of $P(t)$
10. mutate lower module of $P(t)$
11. goto 3
12. output best and stop

So the path planning problem is likely to be solved by HGA. Actually, we use the algorithm above to solve the problem. The following section describes the details of the algorithm.

6.5 Methodology

We will divide this section into the following subsections:

- (1) Relationships between two consecutive circles tangential with the path,
- (2) Fitness Function,
- (3) Criterion for Interception between a Line and a circle with two points,
- (4) Penalty strategy,
- (5) The encoding of chromosomes,
- (6) Genetic operators.

We analyze the relationships between two consecutive circles tangential with the path because it is very basic in the calculation of fitness function. The criterion for interception between a line and a circle with two points is basic for calculating the penalty.

6.5.1 A Segment of a Path

To calculate the length of the path, the length of every segment in a path should be known. Fig. 6-6 shows four types of segment in a “tight path”. The point robot meets the obstacle circle O_1 first, and may travel a segment on circle O_1 , then travel through line P_1P_2 to circle O_2 , and also may travel a segment on circle O_2 . From the discussion, line

Chapter 6 A Path Planning Algorithm

P_1P_2 must be tangential to circle O_1 and circle O_2 . We try to analyze the relationship by the theory of complex variables since it is a good tool to deal with the problem of vectors in a 2D space.

Fig. 6-6 Four types of segment in a path

The center of circle O_1 is (x_1, y_1) , and the center of circle O_2 is (x_2, y_2) .

Let $d = |\overline{O_1O_2}|$, then $\overline{O_1O_2} = (x_2 - x_1) + i(y_2 - y_1)$, and $|\overline{P_1P_2}| = d \sin \alpha$, here $\alpha = \angle P_1O_1O_2$.

There are four relationships as shown in the Fig. 6-5 (a), (b), (c), and (d). Let's analyze the four relationships one by one.

(a) $O_1(+1) O_2 (+1)$ means that the direction of segment on circle O_1 is clockwise (+1)

and the direction of segment on circle O_2 is also clockwise (+1). $\alpha = \arccos \frac{r_1 - r_2}{d}$, and

$$\overline{OP_1} = \overline{OO_1} + \overline{O_1P_1} = (x_1 + iy_1) + r_1 \left[\frac{(x_2 - x_1) + i(y_2 - y_1)}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}} \right] (\cos \alpha + i \sin \alpha) \quad (6-5)$$

$$\overline{OP_2} = \overline{OO_2} + \overline{O_2P_2} = (x_2 + iy_2) + r_2 \left[\frac{(x_1 - x_2) + i(y_1 - y_2)}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}} \right] (-\cos \alpha - i \sin \alpha) \quad (6-6)$$

(b) $O_1(-1) O_2(-1)$ means that the direction of segment on circle O_1 is counterclockwise

(-1) and the direction of segment on circle O_2 is counterclockwise (-1) . $\alpha = \arccos \frac{r_1 - r_2}{d}$,

and

$$\overline{OP_1} = \overline{OO_1} + \overline{O_1P_1} = (x_1 + iy_1) + r_1 \left[\frac{(x_2 - x_1) + i(y_2 - y_1)}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}} \right] (\cos \alpha - i \sin \alpha) \quad (6-7)$$

$$\overline{OP_2} = \overline{OO_2} + \overline{O_2P_2} = (x_2 + iy_2) + r_2 \left[\frac{(x_1 - x_2) + i(y_1 - y_2)}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}} \right] (-\cos \alpha + i \sin \alpha) \quad (6-8)$$

(c) $O_1(+1) O_2(-1)$ means that the direction of segment on circle O_1 is clockwise $(+1)$

and the direction of segment on circle O_2 is counterclockwise (-1) . $\alpha = \arccos \frac{r_1 + r_2}{d}$, and

$$\overline{O_1P_1} = -\overline{O_2P_2} = (x_1 + iy_1) + r_1 \left[\frac{(x_2 - x_1) + i(y_2 - y_1)}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}} \right] (\cos \alpha + i \sin \alpha) \quad (6-9)$$

Chapter 6 A Path Planning Algorithm

(d) $O_1(-1) O_2(+1)$ means that the direction of segment on circle O_1 is counterclockwise

(+1) and the direction of segment on circle O_2 is clockwise (+1). $\alpha = \arccos \frac{r_1 + r_2}{d}$, and

$$\overline{O_1 P_1} = -\overline{O_2 P_2} = (x_1 + iy_1) + r_1 \left[\frac{(x_2 - x_1) + i(y_2 - y_1)}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}} \right] (\cos \alpha - i \sin \alpha) \quad (6-10)$$

6.5.2 Fitness Function

The static path planning is to find the shortest path with obstacle avoidance, or minimize function $Dist(s)$, here s can be any feasible chromosome and $Dist(s)$ is the length of the path represented by chromosome s . Actually, it is the sum of the length of the path which is concatenated by the segments of lines and circles. The section above provides the method for the calculation of the length.

Generally, the GA is to find a chromosome with the **maximum** fitness, so the fitness function can be the reciprocal of the path multiply with a constant, which can be defined as

$$fit(s) = \text{const} / (Dist(s) + 1), \text{ here } \text{const} \text{ is a positive constant.} \quad (6-11)$$

6.5.3 Criterion for Interception

During the evolving process, GA will generate a lot of infeasible paths, which intercepts with one or more obstacles in the field. How could we separate these infeasible paths from the feasible paths? Let us analyze the relationships between a line segment of the path and a circle and try to find out the criterion for interception.

There could be four relationships between a line segment and a circle.

- (1) The line segment intercepts the circle with two common points.
- (2) The line segment intercepts the circle with only one point, which will not happen in this problem.
- (3) The line segment is tangential to the circle, which is a special case in our problem.
- (4) There is no common point between the line segment and the circle.

Condition (1) will not happen for a feasible path. So the problem here is to find out the criterion for condition (1).

Fig. 6-7 Two relationships of a line with a circle.

Fig. 6-7 shows condition (1) and condition (4) between a line and a circle. In Fig. 6-7 (a), a line does not intercept with a circle, which is condition (4) and in Fig. 6-7 (b), a line does not intercept with a circle, which is condition (1). We only need to consider the

Chapter 6 A Path Planning Algorithm

criterion for that *a circle is intercepted by a line with two points*. The theory of complex variables is used here to analyze these relationships.

The equation for a line is

$$S + t(T - S) = Z, 0 \leq t \leq 1. \quad (6-12)$$

Here S is the initial point of the line segment, and T is the terminal point. Both S and T are known complex variables.

The equation for a circle is

$$|Z - Z_0| = R_0, \quad (6-13)$$

There are two solutions in the function if the line segment intercepts the circle. So there should be two solutions in the following equation.

$$|S + t(T - S) - Z_0| = R_0 \quad (6-14)$$

Let $\bar{T} - \bar{S} = x_1 + iy_1$, $\bar{S} - \bar{Z}_0 = x_2 + iy_2$. Obviously, $x_1^2 + y_1^2 \neq 0$.

The above equation can be transformed into

$$(x_1^2 + y_1^2)t^2 + (2x_1x_2 + 2y_1y_2)t + (x_2^2 + y_2^2 - R_0^2) = 0. \quad (6-15)$$

Defining a function

$$f(t) = t^2 + \frac{(2x_1x_2 + 2y_1y_2)}{(x_1^2 + y_1^2)}t + \frac{(x_2^2 + y_2^2 - R_0^2)}{(x_1^2 + y_1^2)}, 0 \leq t \leq 1, \quad (6-16)$$

and let $b = \frac{(x_1x_2 + y_1y_2)}{(x_1^2 + y_1^2)}$, $c = \frac{(x_2^2 + y_2^2 - R_0^2)}{(x_1^2 + y_1^2)}$, then we get

$$f(t) = t^2 + 2bt + c = (t + b)^2 + c - b^2, \quad 0 \leq t \leq 1. \tag{6-17}$$

So the problem is to decide if the function $f(t)$ has two solutions t_1 and t_2 ($t_1 \neq t_2$, $0 \leq t_1, t_2 \leq 1$).

Fig.6-8 The criterion for the function $f(t)$ with two solutions t_1 and t_2

The only condition for function $f(t)$ with two solutions is shown in Fig. 6-8. From this diagram, we can obtain the criterion as shown in Table 6-1.

From the analysis in Table 6-1, the criterion for interception between a line and a circle with two points is

$$(i) \ -1 < b < 0, \ (ii) \ f(-b) < 0, \ \text{and} \ (iii) \ f(0) \geq 0 \text{ and } f(1) \geq 0. \tag{6-18}$$

Table 6-1 Criterion for interception between a line and a circle

Criterion			Solution
-1<b<0	f(-b) ≥ 0		<2
	f(-b)<0	f(0) ≥ 0 and f(1) ≥ 0	=2
		f(0) < 0 and f(1) < 0	<2
b ≥ 0 or b ≤ -1			<2

6.5.4 Penalty Strategy

In the searching process, the GA will produce a lot of infeasible solutions, which could be infeasible paths such as the paths intercepted with the obstacles. Penalty strategy is a common technique to deal with infeasible solutions by a simple idea of “punishing” the infeasible solutions with a penalty.

Since we are going to find the path with the minimum length, the infeasible solution intercepted with the obstacle should be “punished” by adding some extra penalty. The above section provides the criterion for interception between a line and a circle with two points. Since a path is the concatenation of line segments and arcs, we can judge the relationship between each line segment with each circle and count the times of the interception altogether.

Let l_i denote the i th line segment in a possible path, and C_j the j th obstacle, then $Intercept(l_i, C_j)$ equals '1' when C_j is intercepted by l_i , and '0' otherwise.

$\sum Intercept(l_i, C_j)$ is the number of times of interception for a path.

The distance function can be modified into

$$Dist_m(s) = Dist(s) + \delta \cdot \sum Intercept(l_i, C_j) \quad (6-19)$$

Here, δ is a parameter for the penalty, and the perimeter of the circle is selected. Now the fitness function should be

$$fit(s) = \text{const}/(Dist_m(s) + 1) \quad (6-20)$$

6.5.5 The Encoding of the Chromosome

Before we discuss the encoding of the string, we must know four kinds of relationships between two consecutive circles tangential with the tight path in section 6.5.1, which tells us the relationships between two consecutive circles tangential with the path, we also need to analyze the critical points and angles needed in the path, and calculate the length of the path. It is the foundation of rendering the path and the calculation of its length.

In section 6.4.5, we describe the path like $\{0 (-1) 1(+1) 5(+1) 7(-1) 10 (+1)\}$. Fig. 6-9

shows a chromosome representation, $\begin{Bmatrix} 0 & 1 & 5 & 7 & 4 & 2 & 3 & 6 & 8 & 9 & 10 \\ -1 & +1 & +1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & +1 \end{Bmatrix}$, of

the path shown in Fig.6-3 by HGA.

Fig. 6-9 The chromosome represents the path in Fig. 6-3.

This chromosome consists of the *control genes* (which is the upper module) and the *parametric genes* (which is the lower module). The activation of the *parametric gene* is governed by the value of the corresponding *control gene*. The chromosome represents the path in Fig.6-3. The upper module denotes the sequence of the circles encountered, and the lower module denotes the relationships of the path and obstacles. The following is the explanation of the chromosome.

1. The “0” in the first position of the upper module represents the initial position of the moving robot, and controls two selections in the lower module, namely, the left circle (“-1” in the lower module, moving direction is counterclockwise) or the right circle (“+1” in the lower module, moving direction is clockwise) to the initial moving direction.

2. The “10” in the last position of the upper module represents the final position of the moving robot, and controls two selections like the gene “0” in the upper module.
3. The genes from “1” to “9” in the upper module represent the corresponding obstacles denoted in Fig. 6-3, and each gene controls three possible selections as mentioned before: *not tangent* (“0” in the lower module), *tangent and clockwise* (“+1” in the lower module), *tangent and counterclockwise* (“-1” in the lower module).
4. The meaning of the chromosome $\begin{matrix} 0 & 1 & 5 & 7 & 4 & 2 & 3 & 6 & 8 & 9 & 10 \\ -1 & +1 & +1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & +1 \end{matrix}$ is a possible path in the field shown in Fig. 6-3.

From the above explanation, we know how to express a possible path by the encoding mentioned above. But there are a lot of different chromosomes to express the same path. For example, the path mentioned before can be expressed in the chromosome above or in another chromosome $\begin{matrix} 0 & 1 & 5 & 4 & 2 & 3 & 6 & 8 & 9 & 7 & 10 \\ -1 & +1 & +1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & +1 \end{matrix}$, since changing the sequence of the *not tangent* circle(s) does not change the meaning of the chromosome. So a path or the chromosomes with the same meaning can be expressed by the concept of set. In the example above, the path can be represented as

$$\left\{ \begin{matrix} 0 & 1 & 5 & 7 & 4 & 2 & 3 & 6 & 8 & 9 & 10 \\ -1 & +1 & +1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & +1 \end{matrix} \right\}. \quad (6-21)$$

Chapter 6 A Path Planning Algorithm

6.5.6 Genetic operators

After the encoding of the chromosome is known, genetic operators must be selected to search for the optimal chromosome according to the basic steps for HGA mentioned before.

The genetic operators are *selection* operator, *crossover* operator, and *mutation* operator. The selection operator can be *Stochastic Universal Sampling* with *Partial Replacement* with the most common scaling techniques like *Linear scaling*, *Sigma Truncation*, *Power Law Scaling*, or *Ranking*. These operators are very common and can be found in any books about GA [GLDB85].

The crossover and mutation operators will have some changes since the structure of the upper module of the chromosome has its special features while the structure of the lower module of the chromosome is like the structure of the binary chromosome. As a matter of fact, the structure the upper module is much like the structure of chromosome to solve the Traveling Salesman Problem (TSP), so the crossover and mutation operators, which are implemented in TSP, can also be used in crossover and mutation of the upper module with some variances. The crossover and mutation in the lower module is much like the crossover and mutation of the binary GA also with some variances.

Table 6-2 shows the optional crossover and mutation operators for the upper module and the lower module. These genetic operators are very commonly used.

Table 6-2 The crossover and mutation operators

Module	Crossover Operator	Mutation Operator
Upper Module	PMX	Inversion Mutation
	OX	Reciprocal Exchange Mutation
	CX	
Lower Module	Two Points	Common
	Uniform	

For the upper module, *Partially Matched Crossover* (PMX), *Order Crossover* (OX), or *Cycle Crossover* (CX) can be selected as crossover operators, and *Inversion Mutation* or *Reciprocal Exchange Mutation* can be selected as mutation operators. Two points must be considered. Firstly, when applying these operators, the first and last positions in the chromosome are static and do not need to be considered. Secondly, when the sequence of the upper module is changed, so is the sequence of the corresponding parameters in the lower module.

For the lower module, both crossover and mutation operators are quite commonly used, but we must note that because there are only 2 choices for the first and the last positions and 3 choices for the remaining positions, for mutation, there is a little variance. And for crossover, the first position and the last position do not need to be considered for the same reason. See the Appendix E for these operators.

6.6 Simulation Results

We use two examples to illustrate the proposed path planning algorithms. The algorithms were implemented in MATLAB on a PC with a Pentium III 700MHZ CPU, using the features provided above and some other parameters as given below. The population size is 50. The maximum number of generation is 50. The probabilities are selected as follows: For the upper module, $pc = 0.4$, $pm = 0.02$; For the lower module, $pc = 0.4$, $pm = 0.02$. It takes about 5 seconds to complete the search.

Fig. 6-10 The optimal path achieve from HGA.

Fig. 6-10 shows the optimal result from this experiment, which is

$$\begin{Bmatrix} 0 & 1 & 4 & 5 & 7 & 2 & 3 & 6 & 8 & 9 & 10 \\ +1 & -1 & +1 & +1 & -1 & 0 & 0 & 0 & 0 & 0 & +1 \end{Bmatrix}.$$

The fitness over generation is shown in Fig. 6-11. The optimal solution is obtained in generation 27 (assuming that the initial generation number is 1), and a near optimal solution is obtained in generation 11. From the figure, the convergence speed is very fast.

Fig 6-11 Fitness over generation

Because GA is not promised to produce the optimal solution, sometimes some near-optimal solutions are obtained as the one shown in Fig. 6-12, but the probability of obtaining the near-optimal solution is high.

Chapter 6 A Path Planning Algorithm

Fig. 6-12 A near-optimal solution

The second example is illustrated in Fig. 6-13.

Fig. 6-13 Another optimal path generated by HGA

Fig. 6-13 (a) shows the optimal path obtained from HGA, which is

$$\begin{Bmatrix} 0 & 8 & 5 & 4 & 1 & 2 & 3 & 6 & 7 & 9 & 10 \\ +1 & -1 & +1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{Bmatrix}.$$

Fig. 6-13 (b) shows the fitness over generation. The optimal solution is obtained in generation 9 (assuming that the initial generation number is 1). The figure shows that the convergence speed is very fast.

CHAPTER 7

Conclusions and Future Research

This chapter reviews the thesis' contributions to the vision system and the motion planning of robotic soccer in the area of tracking system, calibration, and path planning. It also discusses the directions for future research in this challenging problem.

7.1 Contributions:

The three main contributions of this thesis are summarized as follows.

(1) The tracking system presented in *Chapter 3* is a robust solution for the vision system in the small robot leaguer of the robotic soccer.

In order to control the agents in the field, the Artificial Intelligent Module requires the information of 10 agents and a ball. The task is fulfilled by the visions system. This thesis presents the design and implementation of a real-time, robust global vision system. The work about object tracking include:

- *color-based segmentation* for locating the objects of interest in the image;

- *blob analysis* for calculating the positions of the objects in the images and filtering out the noise;
- *double-check data association* for associating the current time frame information with the previous time frame information;
- *linear filter* for tracking and predicting the states information;
- *UDP socket programming* for transmitting the information from the vision system to the AI system.

(2) The calibration result in *Chapter 4* is based on the global vision system for tracking eleven moving objects.

To obtain the precise motion information of one ball and ten robots, two kinds of calibrations are important: color calibration and camera calibration. This thesis investigates a K-Mean-clustering color calibration algorithm to obtain accurate color identification, and Tsai's camera model to get the calibration parameters for image distortion compensation. Special techniques are implemented to apply these algorithms to achieve the best results.

(3) The path planning result in *Chapter 6* is based on assumption of nonholonomic robot constraints and the exploitation of Dubin's theorem. A novel hierarchical genetic algorithm for path planning in a static environment with obstacles is presented.

Chapter 7 Conclusions and Future Research

Inspired by the Dubin's theorem regarding shortest paths of bounded curvature in the absence of obstacles, this novel path planning algorithm is based on the following knowledge:

- Dubin's theorem to simplify the problem model;
- genetic algorithm to search the best path;
- a special hierarchical structure of chromosome to denote a possible path in the environment;
- the special genetic operators for each module;
- a penalty strategy to "punish" the infeasible chromosomes during searching.

The simulation results presented have shown that the approach is able to produce high quality solutions in reasonable time.

7.2 Recommendations for Future Research

(1) For Vision System

a) Color-based segmentation

Color-based segmentation and blob analysis are two important steps in our vision system. Due to inconsistent lighting conditions, after segmentation, the tracking system will lose some objects' information occasionally even though the tracking system is robust enough. There could be two possible ways to deal with the problem.

One feasible way to improve the performance of segmentation is to use some other color system like HIS instead of RGB. Actually some other teams have already applied this technique [SIMN01].

Techniques based on adaptive thresholding could be another possible way. There are several approaches. One involves modeling the background within the image [DVIS90]. Another is to work out a local threshold value for each pixel by examining the range of intensities in its neighborhood [NBLK85]. A third approach is to split the image into subimages and deal with them independently [CHOW72]. The principles of the second method and the third method can be applied to color-based segmentation. But the third approach involves considerable computation and in real-time applications it may not be viable [DVIS90]. To implement the third approach in color-based segmentation, one can combine this local neighborhood technique with parameters from the clustering algorithm.

b) Speed of the tracking system

Blob analysis is computationally intensive. In real-time applications, it often requires special hardware like DSP or FPGA to fulfill this task in the vision system. In our system, to accelerate the speed of the vision system, there are two ways: one is to use more powerful hardware, and the other one is to optimize the algorithm. We are more interested in the latter solution.

Fortunately, there are ways to optimize the algorithm or to reduce the computational load about blob analysis. One of the most important features in the tracking system is redundant information between the consecutive frames, so we only need to concentrate on a part of information in the current frame.

Chapter 7 Conclusions and Future Research

Based on the knowledge of the Kalman filter, we can predict the range of the objects in the next frames [BRKN88]. If the object is inside the prediction range, then only blobs within the prediction range need to be analyzed, otherwise global search can be applied to locate the lost objects. Actually, according to the filter technology, the probability of the blobs within the prediction range can be very near to 100% [BRKN88]. The commercial product with blob analysis often comes with image labeling, which is also computationally intensive [GNLC00]. Image labeling is not necessary for the tracking system by using the prediction range. One team adopted this technique and it does not need the auxiliary hardware [SIMN01].

c) Camera calibration

The camera calibration in *Chapter 4* still needs human's intervention during calibration. Recently, there are progresses in the area of camera calibration. A new technique, which is called camera self-calibration using the concept of absolute conic [FGRS92] [SMPL52], is more convenient than Tsai's camera calibration [ZHNG00] [ZHNG99]. Compared with traditional calibration techniques, self-calibration does not require a calibration object with known 3D geometry, but only needs point correspondences from image sequences to solve for the intrinsic parameters. It may be applicable in the vision system for robotic soccer.

(2) For Motion Planning

a) Path planning algorithm by HGA

The static path planning algorithm in this thesis is based on HGA. The following are the drawbacks of the algorithm:

- This algorithm is not suitable for real time applications as the speed of this algorithm is not fast if we only implement the algorithm by software.
- As mentioned in section 5.3, we assume each robot is a circle.
- This algorithm is not complete as we mentioned that the shortest path has the form of $X_a X_b X_c X_d \dots X_s X_t$, while we only investigated the form of $C_a S_b C_c S_d \dots S_s C_t$.
- GA is based on probability, which means it is not guaranteed to provide the best solution.

Despite these drawbacks, it is still quite a good algorithm for this difficult problem. How to improve the above algorithm? Because the problem is inherently NP, it is not possible to transform the complexity of the problem within polynomial time according to complexity theory [WKPD02]. However, there are still ways to accelerate the processing speed according to the nature of genetic algorithm. Genetic algorithm is a natural parallel algorithm, and often used to solve NP problems, so parallel structures like global GA, migration GA, or diffusion GA can be implemented into parallel hardware [MANK99]. Parallel hardware implementation of our algorithm may provide better performance.

For the path of $X_a X_b X_c X_d \dots X_s X_t$ in section 6.2, we mentioned that the shortest path could also be found according to similar principle. However, the problem becomes more complicated because we should draw some “virtual” circles of the same radius of

Chapter 7 Conclusions and Future Research

minimal turning radius in the initial model plane with the “real” circles of obstacles. Using the same rope solution” As in the initial model, we can also find the shortest path in the modified model plane.

Calculation of the fitness of chromosomes is computationally intensive in our algorithm. The fitness of a chromosome consists of two operations: interception judgment and computation of path length, which is easier in computation. In our algorithm, we judge the interceptions during evolution. The computational load can be greatly reduced if we judge the interceptions in advance, because there are a lot of repetitive computations during evolution. The information on interceptions can be saved in a table, during evolving we can refer to the table. Some progress in path planning like collision detection and collision avoidance may also be helpful for our algorithm.

b) Some practical algorithms

There are alternative algorithms to solve the problem if some constraints are modified. These algorithms are practical for real-world applications, although they may not be theoretically optimal.

If there is no nonholonomic constraint, and we approximate the obstacles into squares, the path planning can be solved by two approaches based on classical algorithms. One approach is by the grid-based application. The other application is to plan a path globally using A* algorithm and to control the robot with artificial potential fields algorithm. These algorithms were introduced in *Chapter 5*.

c) A possible idea

In *Chapter 6*, we know that a practical path can be seen as a rope between the initial point and the target point, and a feasible path is like a tight rope. From Fig. 6-3, we know that these two paths, a random path and its corresponding feasible path, share the same structure. So one feasible method is to find out a feasible path from a practical path of the same structure. The practical path can be obtained by algorithm like A*. This method has following advantages:

- The computation complexity for a practical path is NP (See 5.2.3 Roadmap Methods).
- A practical path is optimal on the rough conditions, which are not very near to the real conditions. That means the feasible path generated by this method will also be a near-optimal path.

What we have to do is to smooth the practical path and make it feasible for the robot. This method can be viewed as an improvement on the practical algorithms.

d) Comments on motion control in RoboCup

One of the main objectives in RoboCup is to promote motion control, especially dynamic motion planning, yet the concurrent theories in motion planning are still not ready for the complicated situations because the planning in the long term is useless in the complicated environment. Another main objective in RoboCup is to promote Artificial Intelligence, especially in Multi-Agent Systems (MAS), which is the current focus in RoboCup. I

Chapter 7 Conclusions and Future Research

believe that these two objectives are consistent in the long run because MAS is a high-level processing in the AI module while motion control is a low-level processing in the AI module.

Author's Publications:

1 Chunmiao Wang, Hui Wang, William Y. C. Soh, Han Wang, "***A Real Time Vision System for Robotic Soccer***", in Proc. ACRC., 2001, Singapore (*This paper focuses on the calibration techniques.*).

2 Chunmiao Wang, Hui Wang, Han Wang, William Y.C. Soh, "***The Vision System For RoboCup Small Robot League***", in Proc.ASIAR, 2001, Thailand (*This paper focuses on the tracking system.*).

3 Chunmiao Wang, Y. C. Soh, Han Wang, Hui Wang, "***A Hierarchical Genetic Algorithm for Path Planning in a Static Environment with Obstacles***", in WCCI 2002, Hawaii, USA (*This paper provides a new algorithm for path planning.*).

Reference

- [KTNO95] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, and E. Osawa, *RoboCup: The Robot World Cup Initiative*, In Proceedings of the IJCAI-95 Workshop on Entertainment and AI/ALife, 1995.
- [KTNO97] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, E. Osawa, and H. Matsubara, *RoboCup: A Challenge Problem for AI*, AI Magazine 18(1): pp.73-85. 1997.
- [RWCP02] *The Robot World Cup Initiative or RoboCup*, <http://www.robocup.org>, 2002.
- [OBRC02] *Objective of RoboCup*, <http://www.robocup.org/overview/22.html>, 2002.
- [RGLT02] *Regulations and Rules for Small Robot League*, <http://www.robocup.org/regulations/4.html>, 2002
- [NOKH00] N.O. Khessal, *Towards a Distributed Multi-agent System for a Robotic Soccer Team*, RoboCup-99 Robot Soccer World Cup III, Springer Verlag, 2000.
- [BOWL00] M. Bowling and M. Veloso, *Motion Control in Dynamic Multi-Robot Environments*, RoboCup-99 Robot Soccer World Cup III, Springer Verlag, 2000.
- [OSWD00] N. Oswald, M. Becht, T.Buchheim, *CoPS-Team Description*, RoboCup-99 Robot Soccer World Cup III, Springer Verlag, 2000.
- [MORN00] C. Moreno, A. Suarez, Y. Amirat, *Real MagiCol99: Team Description*, RoboCup-99 Robot Soccer World Cup III, Springer Verlag, 2000.
- [ACHM96] S. Achim, P. Stone, and M. Veloso. *Building a Dedicated Robotic Soccer System*, in Proc. of International Conference on Intelligent Robots and Systems 1996 (IROS'96) Workshop on RoboCup.

-
- [WYTH99] G. Wyeth, B. Browning, and A. Tews. *UQ RoboRoos: Preliminary Results for Robot Soccer*, RoboCup-98 Robot Soccer World Cup II, Minoru Asada, Hiroaki Kitano (Eds.), Springer, 1999
- [ANDR00] R. D' Andrea, J.W. Lee, A. Hoffman, A. Samad-Yahaya, L.B. Cremean, and T. Karpati. *Big Red: The Cornell Small League Robot Soccer Team*, RoboCup-99 Robot Soccer World Cup III, Manuela Veloso, Enrico Pagello, Hiroaki Kitano (Eds.), Springer, 2000.
- [GNZL92] R.C. Gonzalez and R.E. Woods, *Digital Image Processing*, Addison-Wesley, 1992.
- [GNLC00] *Matrox Genesis Native Library Version 2.1 -- Command Reference*, Manual no. 10495-701-0210, May 12, 2000.
- [BRKN88] E. Brookner, *Tracking and Kalman Filtering Made Easy*, John Wiley & Sons, 1988.
- [TNBM96] A.S. Tanenbaum, *Computer Networks*, 3rd ed, Prentice Hall, 1996.
- [STVN94] R. Stevens, *TCP/IP Illustrated, Volume I: The protocols*, Addison Wesley, 1994.
- [SIMN01] M. Simon, S. Behnke, and R. Rojas, *Robust Real Time Color Tracking*, RoboCup 2000 Robot Soccer World Cup IV, Springer, 2001.
- [HOCH00] M. Hoch and S. Mann, *Fuzzy-Based Person Tracking in Real-Time*, Computer Graphics International, pp201-208, 2000.
- [VLSO98] M. Veloso, P. Stone, K. Han, and S. Achim, *The CMUnited-97 Small Robot Team*, RoboCup-97: Robot Soccer World Cup I, H. Kitano (ed.), Springer Verlag, Berlin, 1998.

Reference

- [DUDA73] R.O. Duda, and P.E. Hart, 1973, *Pattern Classification and Scene Analysis*, John Wiley & Sons, New-York.
- [MCQN67] J. MacQueen, *Some Methods for Classification and Analysis of Multivariate Observations*, Proceeding of the Fifth Berkeley Symposium on Mathematics, Statistics and Probability, pp281-296, 1(1967)
- [ANDB73] M.R. Anderberg, *Cluster Analysis for Application*, Academic Press, New York, 1973.
- [HRTG75] J.A. Hartigan, *Clustering Algorithms*, Wiley, New York, 1975.
- [DVJV82] P.A. Devijver, J. Kittler, *Pattern Recognition: A Statistical Approach*, Prentice-Hall, London, 1982.
- [JAIN88] A.K. Jain, R.C. Dubes, *Algorithms for Clustering Data*, Prentice-Hall, Englewood Cliffs, NJ, 1988.
- [TOUT74] J.T. Tou, R.C. Gonzalez, *Pattern Recognition Principles*, Addison-Wesley, Reading, 1974.
- [SELM87] S.Z. Selim, M.A. Ismail, *K-means type algorithms: a generalized convergence theorem and characterization of local optimality*, IEEE Trans. Pattern Anal. Mach. Intell., pp 81-87, 6(1984).
- [TSAI86] R.Y. Tsai, *An Efficient and Accurate Camera Calibration Technique for 3D Machine Vision*, Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Miami Beach, FL, 1986, pages 364-374,
- [TSAI87] R.Y. Tsai, *A versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses*, IEEE Journal of Robotics and Automation, Vol. RA-3, No. 4, August 1987, pages 323-344.

-
- [LTMB91] J.C. Latombe, *Robot Motion Planning*, Kluwer Academic Publishers, 1991.
- [PARK99] L.E. Parker, J.V. Draper, *Handbook of Industrial Robotics*, Second Edition, Shimon Nof (ed.), 1999
- [FORN88] P. Fiorini, Z. Shiller, *Motion Planning in Dynamic Environments Using Velocity Obstacles*, International Journal of Robotics Research, July 1998, Vol. 17, No. 7, pp. 760-772. 1988.
- [STON98] P. Stone, *Layered Learning in Multi-Agent Systems*, {AAAI}/{IAAI}, <http://citeseer.nj.nec.com/article/stone98layered.html>, 1997
- [STON00] P. Stone, *Layered Learning in Multi-Agent Systems: A Winning Approach to Robotic Soccer*, MIT Press, 2000.
- [CANN87] Canny and Reif, *New Lower Bound Techniques for Robot Motion Planning Problems*, In 28th Symposium on Foundation of Computer Science, 1987.
- [SNBN88] Sanborn and Hendler, *A Model of Reaction for Planning in Dynamic Environments*. International Journal of Artificial Intelligence in Engineering, 3(2): pp95-101, 1988.
- [LMND86] J.P. Laumond, *Feasible Trajectories for Mobile Robots with Kinematic and Environment Constraints*, Preprints of the International Conference on Intelligent Autonomous Systems, Elsevier Science Publishers B.V., Amsterdam, The Netherlands, 346-354, 1986
- [LMND87] J.P. Laumond, *Find Collision-Free Smooth Trajectories for a Non-holonomic Mobile Robot*, Proceedings of the 10th International Joint Conference on Artificial Intelligent, Milan, Italy, pp1120-1123, 1987

Reference

- [BRQN89] J. Barraquand, and J.C. Latombe, *On Non-Holonomic Mobile Robots and Optimal Maneuvering*, Revue d'Intelligent Artificielle, 3(2), Hermes, Paris, pp77-103, 1989
- [CYLE61] C.Y. Lee, *An Algorithm for Path Connections and Its Applications*. IRE Trans. Electronic Computers, pages 346--365, September 1961.
- [JRVS83] R. A. Jarvis, *A Perspective on Range Finding Techniques for Computer Vision*, IEEE Transactions on Pattern Analysis and Machine Intelligence. Vol. PAMI-5, No.2., March 1983, pp.122-139.
- [JRVS88] R. A. Jarvis, J. C. Byrne and K. Ajay, *An Intelligent Autonomous Guided Vehicle: Localisation, Environmental Modelling and Collision Free Path Finding*, Proceedings of 19th ISIR, International Symposium and Exposition of Robotics , pp.767-792, Sydney, Australia, Nov. 1988.
- [KHTB86] O. Khatib, *Real-Time Obstacle Avoidance for Manipulators and Mobile Robots*, International Journal of Robotics Research, 5(1), pp. 90-98. 1986.
- [NSSN69] N.J. Nilsson, *A Mobile Automation: An Application of Artificial Intelligence Techniques*, Proceeding of the 1st International Joint Conference on Artificial Intelligence, Washington, D.C., pp509-520, 1969.
- [NSSN80] N.J. Nilsson, *Principles of Artificial Intelligence*, Morgan Kaufmann, Los Altos, CA, 1980.
- [ODNL83] C. O'Dunlaing, M. Sharir, and C.K. Yap, *Retraction: A New Approach to Motion Planning*, Proceedings of the 15th ACM Symposium on the Theory of Computing, Boston, pp207-220, 1983.

-
- [ODNL84] C. O'Dunlaing, M. Sharir, and C.K. Yap, Retraction: *Generalized Voronoi Diagrams for Moving a Ladder: II. Efficient Construction of the Diagram*, Technical Report No. 32, Robotics Laboratory, Courant Institute, New York University, 1984
- [AMTO96] N. Amato and Y. Wu. *A Randomized Roadmap Method for Path and Manipulation Planning*, In Proc. IEEE Int. Conf. on Robotics and Automation, pages 113--120, 1996.
- [LZPR79] T. Lozano-Perez and M. Wesley, *An Algorithm for Planning Collision-free Paths Among Polyhedral Obstacles*, Communications of the ACM, 22, 1979,
- [LEED78] D.T. Lee, *Proximity and Reachability in the Plane*, Ph.D. Dissertation, Coordinated Science Laboratory, University of Illinois, Urbana, IL, 1978
- [WLZL85] E. Welzl, *Constructing the Visibility Graph for n line Segments in Time*, Information Processing Letters, 20, pp167-171, 1985
- [GHSH87] S.K. Ghosh and D.M. Mount, *An Output Sensitive Algorithm for Computing Visibility Graphs*, Proceedings of the 28th IEEE Symposium on Foundation of Computer Science, Los Angeles, 11-19, 1987,
- [DUBN57] L.E. Dubins, *On Curves of Minimal Length with A Constraint on Average Curvature and with Prescribed Initial and Terminal Positions and Terminal and Tangents*, American Journal of Mathematics, Vol. 79, pp. 497-516, 1957.
- [LMND98] J.P. Laumond, *Robot Motion Planning and Control*, Laboratoire d'Analyse et d'Architecture des Systèmes Centre National de la Recherche Scientifique LAAS report 97438, Previously published as: Lectures Notes in Control and Information Sciences 229. Springer, ISBN 3-540-76219-1, 1998, 343p.
- [TSPH02] Math Department of Princeton, *History of TSP*,

Reference

<http://www.math.princeton.edu/tsp/>, 2002

[MSCT89] P. Moscato, *On Genetic Crossover Operators for Relative Order Preservation*, Caltech Concurrent Computation Program, C3P Report 778, 1989.

[WHTL92] D. Whitley and K. Mathias, *Genetic Operators, the Fitness Landscape and the Traveling Salesman Problem*, Parallel Problem Solving from Nature-PPSN 2. R. Manner and B. Manderick, eds., pp. 219-228. North Holland-Elsevier, 1992

[DZBR94] J. Dzuberka and D. Whitley, *Advanced Correlation Analysis of Operators for the Traveling Salesman Problems*, Parallel Problem Solving from Nature-PPSN III. Y. Davidor, H.P. Schwefel and R. Manner, eds. pp. 68-77. Springer-Verlag, 1994.

[HLLD75] J.K. Holland, *Adaption in Natural and Artificial Systems*, MIT Press, 1975

[DVIS85] L. Davis, *Job Shop Scheduling with Genetic Algorithms*, Proc. 1st Int. Conf. Genetic Algorithms, J.J. Grefenstette (Eds), 136-140, 1985

[GLDB85] D.E. Goldberg and R. Linge, *Alleles, Locis, and the Traveling Salesman Problem*, Proc. Int. Conf. Genetic Algorithms and Their Applications, 154-159, 1985

[SMTH85] D. Smith, *Bin Packing with Adaptive Search*, Proc. Int. Conf. Genetic Algorithms and Their Applications, 202-206, 1985

[GNLU98] *Matrox Genesis Native Library ver.1.2 – User Guide*, Manual no. 10482-MU-0120, February 10, 1998,

[DVIS90] E.R. Davies, *Machine Vision: Theory, Algorithms, Practicalities*, The USA: Academic Press Inc., 1990.

[CHOW72] C.K. Chow and T. Taneko, *Automation Boundary Detection of the Left Ventricle from Cineangiograms*, Computer and Biomedical Research 5, pp.388-410, 1972.

[NBLK85] W. Niblack, *An introduction to Digital Image Processing*, Strandberg, Birkerød, Denmark, 1985.

[FGRS92] O. Faugeras, Q.T. Luong and S. Maybank, *Camera Self-Calibration: Theory and Experiments*, in Proceedings of the 2th European Conference on Computer Vision, pp.321-334, Italy, 1992.

[SMPL52] J.G. Semple and G.T. Kneebone, *Algebraic Projective Geometry*, Oxford: Clarendon Press, 1952, Reprinted 1979.

[ZHNG99] Z. Zhang, *Flexible New Technique for Camera Calibration*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(11), pp.1330-1334, 2000.

[WKPD02] Wikipedia, *the Free Encyclopedia*, <http://www.wikipedia.com/wiki/NP>. 2002.

[MANK99] K.F. Man, K.S. Tang and S. Kwong, *Genetic Algorithms: Concepts and Designs*, Springer-Verlag London Limited, 1999.

Appendix A -- The Rules of Small Size Team

These are the official rules and regulations from robocup official website.

1 Field Size

A table tennis table is used as the pitch for the official match. The size and colour of the table is officially determined as the international standard for table tennis. It is 152.5cm by 274cm, and color is green. Details shall be given in the figure. Every effort shall be made to ensure that the table is flat, however, it is up to individual teams to design there robots to cope with slight curvatures of the surface.

Four small panels are attached to the corners to avoid balls getting stuck. As shown in the figure below, it is located 3 cm from the corner for each axis. Green strips of width 1cm are painted on the corners to identify the edge of the panel.

For details of the table tennis table, please refer to the *International Table Tennis Federation Regulations*. The regulation concerning the height of the table do not apply. We will provide an appropriate height for the RoboCup competition, which will be chosen to aid the global vision systems.

2 Robot

The total floor area occupied by a robot should not be more than 180 square centimeters, and the maximum length of the body shall not be more than 18centimeters. Height of the robot, if the team is using a global-vision system is restricted to less than 15 centimeters,

otherwise the robot height is 22.5 centimeters. If a robot is to exceed the size for any reason advice should be sought from the co-chairs before the competition. The robots will be examined by the referee before the game to ensure that it fulfills this requirement. Whilst being inspected the robot must be at its maximum size, anything which protrudes from the robot must be extended.

3 Team

A team should consist of no more than 5 robots.

4 Goals

The width of the goal is 50 cm, which is approximately $\frac{1}{3}$ of the length of the shorter end of the field. The goal is 18 centimeters deep. The 10 centimeter wall continues behind the goal, and there is no safety net over the goal. Robots can not fall off the playing area because of the wall. The wall and area behind the goal line will be coloured either dark blue or dark yellow, as will the walls around the goal. It should be noted that a robot may use the area behind the goal.

5 Ball

Orange golf ball shall be used.

6 Colorings

Colors of each part of the field are as follows:

Appendix A -- The Rules of Small Size Team

Field shall be dark green. Since ITTF's color regulation is flexible, there may be slight difference in the color of the table. The designer of the robot shall take this fact in consideration.

- Walls are white.
- Ball is orange.
- The behind goal area is either dark blue or yellow (one end is dark blue and the other yellow).
- Lines are drawn in white.
- Some markers on corners and goals.
- Markers on the robot as specified below.

Coloring is a very sensitive issue, because it is very difficult to guarantee exact color which will be used at the competition. It also changes depends on lighting. While the organizer will make a sincere effort to provide standard color, the designer shall design their robot robust enough to cope with variations in color. Each team will be given certain amount of time to fine-tune their robot for the actual fields and settings on a day before the competition.

Markings for robots need to enable visibility from above (for global vision) and from the playing field (for mobile vision). To support this, robots will be marked using colored ping-pong balls mounted on their top surfaces placed so they do not touch. Unless the shape of the robot does not allow it the two markers must lie on the axis of movement. If this is not possible the relationship between the marker placements and the axis of

movement must be advertised before the competition. One of the markers will be specified as either yellow or dark blue (at the start of each game by the referee). Each team must be able to use either color as the primary color. The second marker can be any color provided the team has registered the colors to be used before the start of the competition. The color can be different from one robot to the next.

For mounting purposes, ping pong balls will be drilled with two small holes (approx 1-2 mm) through their axis to provide for mounting on a spindle. Each robot will be fitted with two spindles of equal height above the ground for holding the balls.

7 Table markings

A 1 centimeter thick white line will be painted across the table (the center line), with a center circle 25 centimeters in diameter placed in the center of the table.

8 Length of the game

The games consists of the first half, break, and the second half. Each of them is 10 minutes. Each time will be allowed some set up time at the start of the game.

9 Wall

A wall shall be placed all around the field, including behind the goals. The wall shall be painted in white and shall be 10 centimeters high. Behind the goals it will be painted in the colours as already described.

Appendix A -- The Rules of Small Size Team

10 Defense Zone

A defense zone is created around each of the goals. It is 22.5 centimeters from the goal line and 100 centimeters wide. The boarder of the defense zone will be painted in white, with the width of 1cm. Only one robot from each team can enter this area. A brief passing and accidental entry of other robots are permitted, but intensional entry and stay is prohibited. Once a defending robot (goal keeper) has hold of the ball or is facing and in contact with the ball then the attacking robot must leave the area. The attacking robot can not interfere with the goal keeper. Given the size of the defense zone a robot is said to be in the defense zone if any part of it is within the area. Also, an attack robot can not intentionally interfere with the movement of the defenders robot in the defense zone. A robot can not be used to block the movement of the goal keeper.

11 Wireless Communications

Robot can use wireless communication to computers and network located outside of the field. The method of wireless communication, power, and frequency shall be notified at the time of registration (by the 1st of May). Any change after the registration shall be notified to the tournament committee as soon as possible. In order to avoid interference, a team should be able to select two carrier frequencies before the match. The type of the wireless communication shall follow legal regulations of the country where the competition is held.

12 Global Vision System / External Distributed Vision System

The use of a global vision system and an external distributed vision system is permitted, not required, to identify and track the position of robots and balls. This is achieved by using one or more cameras. The height of the beam will be at least 2 meters above the table. The placement of cameras is performed on a game by game basis, and the teams choose camera positions by tossing a coin to find which team places a camera first. The use of the global vision system shall be notified at the time of register, and detailed arrangements shall be discussed with the RoboCup organizing committee. The local organiser will inform all participants of the camera attachments required to use the beam provided.

13 Lighting

Separately defined strength of lighting will be provided. This is currently said to be between 700-1000 LUX uniform light.

14 Goal Keepers

Goal keeper can hold and manipulate a ball for up to 10 seconds within its penalty area. After releasing the ball, the keeper must not recapture the ball until it touches any opponent, or a member of its own team outside the penalty area. If the ball is released by the keeper and it reaches the half way line of without touching any other robots, the opponent is given an indirect free kick positioned anywhere along the half way line (borrowed from Futsal rule).

15 During play

Appendix A -- The Rules of Small Size Team

Substitutions can not occur during play, unless a robot is damaged. If a robot is damaged the referee will stop play and the team will have no more the 5 minutes to remove, repair or replace the damaged robot.

- All time for stoppages will be added to the end of the half they occur in.
- The ball has to go forwards at a kick-off or restart.
- In general hand movement of robots is not acceptable. For teams not using a global vision system at kick-offs and restarts a member of the team may allow one person on the pitch to place robots.
- The ball may be lifted during play. However, the height of the ball from the table must not endanger spectators, the referees or human team members!

16 Kick-off/Restart/Stop

For the kick-off, restart, or stoppage (penalty, free kick etc) of the game, the umpire will call verbally, or by whistle, and the operator of the team can send signals to robots. The signal can be entered through a keyboard attached to a server being used on the side lines.

17 Robot positions at kick-off/Restart

Kick-off All robots shall be in located in their side of the field. Only one robot from the side to kick the ball shall be in the center circle.

Penalty Kick Only a goal keeper shall be in the defense zone, and the ball shall be located at the specified position (as determined by the referee). All other robots shall be

located at least 15 cm behind the robot which kicks the ball. Robots other than the goal keeper and the kicker cannot move until the kicker kicks the ball.

Free Kick Currently, all free kicks, or restarts after other stoppages will take the form of a kick-off. If a team has been fouled then the team will be awarded the free kick otherwise the team last to have touched the ball will have the kick off.

Restart after the goal The restart after the goal shall adopt the same formation as the kick-off.

Throw-in When the ball fell-off from the wall, the ball will be returned immediately to the field, and located approximately 5cm inside of the wall, where the ball fell-off. No other robot shall enter 15 cm from the ball. A robot of the team which did not push the ball out of play will be located in front of the ball. During this period, the robot can continue to move, and the time counting continues.

Lack of progress If it is deemed by the referee that the game has stopped then a free kick is awarded to the team which last touched the ball. A game is considered stopped if that ball has not been touched by a robot for 30 seconds and it appears that no robots are likely to hit the ball.

18 Fouls

Following fouls are defined:

Appendix A -- The Rules of Small Size Team

Multiple Defense When more than one robot of the defending side enters the defense zone to substantially affects the game. The foul will be called, and the penalty kick will be declared.

Ball Holding A player cannot 'hold' a ball unless it is a goal keeper in its penalty area. Holding a ball means taking a full control of the ball by removing its entire degrees of freedom; typically, fixing a ball to the body or surrounding a ball using the body to prevent accesses by others. In general 80% of the ball should be outside a convex hull around the robot. This is up to the referee to judge whether a robot is holding the ball. In general another robot should be able to remove the ball from another player. If a robot is deemed to be holding the ball then a free kick will be declared. If this happens in the defense zone by the defense team, a penalty kick will be declared.

Court Modification Modification or damage to the court and the ball is forbidden. Should this occur, the game is suspended and the appropriate restoration is done immediately before the game resumes.

Robot Halting All the players must be halted prior to kick-off or restarting of the game. The judges check or adjust the placements of the players and declares the completion of adjustment 5 seconds before indicating a kick-off or a restart action. During this 5 seconds, the players can not move.

Offside Off side rule is not adopted.

19 Charging

Unless during striving for a ball, a player must not attack another. In case the umpire clearly observes such an act, it is regarded as a violent action. Then the umpire presents a

red card to the responsible player ordering it to leave the game. The judgment is done based on an external appearance. In general, it is unacceptable for multiple robots to charge a single robot, and it is also unacceptable to hit the back of a robot even if it has the ball and it is unacceptable to push along the table another player. The exact interpretation of what is acceptable is left to the referee.

Throughout the game, if a player utilizes a device or an action which continuously exerts, or whose primal purpose appears to be, serious damages to other robot's functions, the umpire can present a yellow card as a warning to the responsible player, and order it to go outside the court and correct the problem. Once the correction is made, the robot can resume to the game under an approval by the umpire. In case the problem is repeated, the umpire presents a red card to the responsible player telling it to leave the game. This rule could be invoked on a robot should it continuously charge a robot whilst attempting to tackle the robot.

Aside from the above items, no regulations are placed against possible body contacts, charging, dangerous plays, obstructions etc. However, it is expected that the aim of all teams is to play a fair and clean game of football.

20 Conflict Resolution

Resolution of dispute and interpretation of ambiguity of rules shall be made by three officials, who will act as umpires, designated prior to the match. The umpires shall not have any conflict of interest to teams in the match. The umpires may consult with the

Appendix A -- The Rules of Small Size Team

tournament officials of the RoboCup for resolving conflicts. Ambiguities shall be resolved by referring to FIFA official regulations, where appropriate. Specific modifications to the rules to allow for special problems and/or capabilities of a team's robots may be agreed to at the time of the competition, provided a majority of the contestants agree.

Appendix B -- Matrix DSP Card

A vision card is needed to communicate between the camera and the computer. In this case, Matrox Genesis board [GNLUSER98] was selected in this project. The board with a transfer rate of 60 Hz has been tested and verified stable with a CCD camera.

The board is equipped with Texas Instruments TM320C80 ('C80) 32-bit RISC master processor (MP), which contains four parallel processors (PP), which are 32-bit fixed point DSPs perform tremendously fast with 64 MB of SRAM. Matrox is among the best image-processing product in the world.

The Matrox genesis board also included an extensive library of image processing functions. The library, Matrox Genesis Native Library 2.1 is adapted extensively in this project [GNLC00]. The time consuming and exhaustive image-processing and blob analysis tasks are implemented on this board (image processing module). On the other hand, the orientation, identification and tracking tasks are done on the host computer (tracking module). The two modules are executed in parallel to achieve real time effect.

With the use of Matrox Genesis vision board and the CV-M70 camera, a Pentium® 500 MHz computer running on WinNT platform is necessary. Both manufacturers recommended the operating system as well. Aside from that, the operating system was proven stable and high security feature was included.

Appendix B -- Matrix DSP Card

With the highly sophisticated technologies, the objectives of the project seem to be achievable. There is only one more step to meet the goals, a good software algorithm.

Appendix C – DXC-9000 Camera with Zoom Lens

DXC-9000 CCD progressive scan full frame colour camera was chosen in this project. The camera is operating at 60 Hz on non-interlace signal per seconds. With a resolution of 640 x 480 (700 TV lines) and SNR of 58 dB, the camera is able to update images at a rate of 60 frames per second. A frame rate of 60 Hz is sufficient in global vision architecture.

As stated in the rules and regulations of *RoboCup*, the camera should be mounted at a height of 3 meters or lower above the playing field. However, the maximum false ceiling height in the laboratory is only 2.3 meters. Therefore, a camera pole with adjustable height from 2 meters to maximum of 3.3 meters is designed and fabricated. The camera was mounted on the pole above field. Consequently, the camera height is adjustable easily.

However, the use of camera has two drawbacks. Firstly, the image acquired by the camera is contaminated by the flickering effect of the fluorescent lights installed in the laboratory.

Secondly, the camera lens caused “fish-eye” distortions (barrel distortion) to the image. This is the inherent measurement error of a camera. Accurate prediction of moving objects’ future location in the image becomes more complicated unless the distortion is corrected with some algorithms.

Appendix D – Data for DXC-9000 Calibration

1. cc_cd.dat ($z_w=0$)

x_w	y_w	X_f	Y_f
3.5523474121e+002	1.2791168213e+002	5.1000000000e+001	4.4000000000e+001
3.5651519775e+002	3.7661538696e+002	5.1084747314e+001	7.0220336914e+001
3.5779565430e+002	6.2531909180e+002	5.1401786804e+001	9.6937500000e+001
3.5907614136e+002	8.7402282715e+002	5.1958332062e+001	1.2304166412e+002
3.6035659790e+002	1.1227265625e+003	5.2041667938e+001	1.4995832825e+002
3.6163705444e+002	1.3714301758e+003	5.2500000000e+001	1.7600000000e+002
6.7507775879e+002	1.2900569153e+002	8.4000000000e+001	4.3000000000e+001
6.7625689697e+002	3.7770941162e+002	8.4233871460e+001	6.9008064270e+001
6.7743603516e+002	6.2641308594e+002	8.4041664124e+001	9.6041664124e+001
6.7861523438e+002	8.7511682129e+002	8.5000000000e+001	1.2300000000e+002
6.7979443359e+002	1.1238205566e+003	8.5000000000e+001	1.5000000000e+002
6.8097357178e+002	1.3725241699e+003	8.5033897400e+001	1.7650000000e+002
9.9492077637e+002	1.3009971619e+002	1.1800000000e+002	4.2000000000e+001
9.9599859619e+002	3.7880340576e+002	1.1850000000e+002	6.9000000000e+001
9.9707647705e+002	6.2750714111e+002	1.1845801544e+002	9.5961830139e+001
9.9815435791e+002	8.7621081543e+002	1.1850757599e+002	1.2308333588e+002
9.9923223877e+002	1.1249145508e+003	1.1906504059e+002	1.5009756470e+002
1.0003100586e+003	1.3736181641e+003	1.1909756470e+002	1.7694308472e+002
1.3147636719e+003	1.3119372559e+002	1.5300000000e+002	4.2000000000e+001
1.3157402344e+003	3.7989743042e+002	1.5267187500e+002	6.9109375000e+001
1.3167167969e+003	6.2860113525e+002	1.5296720886e+002	9.6049179077e+001
1.3176933594e+003	8.7730487061e+002	1.5357554626e+002	1.2369784546e+002
1.3186699219e+003	1.1260085449e+003	1.5354199219e+002	1.5096183777e+002
1.3196466064e+003	1.3747122803e+003	1.5400000000e+002	1.7800000000e+002
1.6346066895e+003	1.3228775024e+002	1.8753846741e+002	4.1461540222e+001
1.6354819336e+003	3.8099145508e+002	1.8800000000e+002	6.9000000000e+001
1.6363572998e+003	6.2969519043e+002	1.8850000000e+002	9.6000000000e+001
1.6372325439e+003	8.7839886475e+002	1.8853900146e+002	1.2353900909e+002
1.6381079102e+003	1.1271025391e+003	1.8850000000e+002	1.5100000000e+002
1.6389831543e+003	1.3758062744e+003	1.8867187500e+002	1.7796093750e+002
1.9544497070e+003	1.3338175964e+002	2.2250000000e+002	4.2000000000e+001
1.9552236328e+003	3.8208547974e+002	2.2300000000e+002	6.9000000000e+001
1.9559976807e+003	6.3078918457e+002	2.2325000000e+002	9.5992187500e+001
1.9567716064e+003	8.7949291992e+002	2.2355555725e+002	1.2370370483e+002
1.9575456543e+003	1.1281965332e+003	2.2345800781e+002	1.5096183777e+002
1.9583195801e+003	1.3769002686e+003	2.2304917908e+002	1.7795901489e+002
2.2742924805e+003	1.3447578430e+002	2.5750000000e+002	4.3000000000e+001

2.2749653320e+003	3.8317950439e+002	2.5719198608e+002	6.9944000244e+001
2.2756381836e+003	6.3188317871e+002	2.5801550293e+002	9.6542633057e+001
2.2763107910e+003	8.8058691406e+002	2.5800000000e+002	1.2400000000e+002
2.2769833984e+003	1.1292905273e+003	2.5790908813e+002	1.5099172974e+002
2.2776562500e+003	1.3779942627e+003	2.5709323120e+002	1.7721186829e+002
2.5941357422e+003	1.3556979370e+002	2.9173504639e+002	4.3786323547e+001
2.5947070313e+003	3.8427349854e+002	2.9100000000e+002	7.0000000000e+001
2.5952783203e+003	6.3297723389e+002	2.9200000000e+002	9.7000000000e+001
2.5958498535e+003	8.8168090820e+002	2.9200000000e+002	1.2400000000e+002
2.5964211426e+003	1.1303846436e+003	2.9150000000e+002	1.5100000000e+002
2.5969926758e+003	1.3790883789e+003	2.9110620117e+002	1.7735398865e+002

2. cc_cpcc.dat

```

Ncx; /* [sel] Number of sensor elements in camera's x direction */ 3.5000000000e+002
Nfx; /* [pix] Number of pixels in frame grabber's x direction */ 3.5000000000e+002
dx; /* [mm/sel] X dimension of camera's sensor element (in mm) */ 1.9800000000e-002
dy; /* [mm/sel] Y dimension of camera's sensor element (in mm) */ 1.9800000000e-002
dpx; /* [mm/pix] Effective X dimension of pixel in frame grabber */ 1.9800000000e-002
dpy; /* [mm/pix] Effective Y dimension of pixel in frame grabber */ 1.9800000000e-002
Cx; /* [pix] Z axis intercept of camera coordinate system */ 1.6194171242e+002
Cy; /* [pix] Z axis intercept of camera coordinate system */ 1.3065723576e+002
sx; /* [] Scale factor to compensate for any error in dpx */ 1.0000000000e+000
f; /* [mm] */ 1.1626336538e+001
kappa1; /* [1/mm^2] */ 3.9448324144e-003
Tx; /* [mm] */ -1.3962699769e+003
Ty; /* [mm] */ -9.4305889256e+002
Tz; /* [mm] */ 5.3684157086e+003
Rx; /* [rad] */ 5.8092292767e-002
Ry; /* [rad] */ 5.4888103539e-002
Rz; /* [rad] */ 7.0769973963e-004
p1; /* [1/mm] */ 0.0000000000e+000
p2; /* [1/mm] */ 0.0000000000e+000

```

Appendix E – Some TSP Operators

The following operators are commonly used in TSP problem, yet not very common for the ordinary GA.

1 Crossover Operators

1.1 Partially Matched Crossover (PMX)

PMX can be viewed as an extension of two-point crossover to representation. The procedure is as follows:

- 1. Choose two crossing sites uniformly at random along the strings.
- 2. Exchange the segments as in two-point crossover.
- 3. Identify the mapping relationship to “repair” the two strings.
- 4. Apply the mapping relationship to “repair” the two strings.

The procedure is illustrated in the following example.

- 1)

9	8	4		5	6	7		1	3	2	10
8	7	1		2	3	10		9	5	4	6
- 2)

9	8	4		2	3	10		1	3	2	10
8	7	1		5	6	7		9	5	4	6
- 3) $5 \Leftrightarrow 2, 6 \Leftrightarrow 3, 7 \Leftrightarrow 10$
- 4)

9	8	4		2	3	10		1	6	5	7
8	10	1		5	6	7		9	2	4	3

1.2 Order Crossover (OX)

1. Choose two crossing sites uniformly at random along the strings.
2. Copy the substrings into the children, substring 1 to child 2 and vice-versa, as in two-point crossover.
3. In the parents, delete those cities which match the cities in the mates's substring which has been copied to the children.
4. Copy the remaining cities, from left to right, from parent 1 to child 1. Similarly for parent 2 to child 2.

We apply this procedure to the previous example.

- 1)

9	8	4		5	6	7		1	3	2	10
8	7	1		2	3	10		9	5	4	6
- 2)

Child1	-	-	-		2	3	10		-	-	-
Child2	-	-	-		5	6	7		-	-	-
- 3)

Parent1	9	8	4		5	6	7		1	-	-
Parent2	8	-	1		2	3	10		9	-	4
- 4)

Child1	9	8	4		2	3	10		5	6	7
Child2	8	1	2		5	6	7		3	10	9

1.3 Cycle Crossover (CX)

Cycle crossover (CX) takes some cities from one parent and selects the remaining cities from the other parent. The cities from the first parent are not chosen at random and, starting from the first city, only those cities which form a cycle according to the corresponding positions between the parents are selected. The procedure is:

1. Find the cycle which is defined by the corresponding positions of cities between parents.

Appendix E – Some TSP Operators

2. Copy the cities in the cycle to a child with the corresponding positions of one parent.
3. Fill the remaining cities by taking them from the other parents.

Suppose we start with the two strings:

9	8	2	1	7	4	5	10	6	3
1	2	3	4	5	6	7	8	9	10

For the first child, we choose 9 from the first string which corresponds to 1 in the second string. Choose 1 from the first string, followed by 4 and 6, this completes the cycle as 6 leads back to 9. This completes steps 1 and 2 for the first child. A similar process for the second child leads to:

9	-	-	1	-	4	-	-	6	-
1	-	-	4	-	6	-	-	9	-

In step 3 we now fill in the remaining cities for the first child from the second parent and vice-versa. This gives:

9	2	3	1	5	4	7	8	6	10
1	8	2	4	7	6	5	10	9	3

2 Mutation Operators

2.1 Inversion Mutation

Suppose there is the original string is $\begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \end{matrix}$, and the two inversion sites have been chosen as 2 and 6 at random within the string, then inverts the substring between these two positions and $\begin{matrix} 1 & 2 & 6 & 5 & 4 & 3 & 7 & 8 & 9 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \end{matrix}$ is the string after transformation.

2.2 Reciprocal Exchange Mutation

Two cities are chosen random and interchanged. This can also be a useful mechanism for reducing crossovers in a tour.

Appendix F. Linear Filter

Our system adopt linear filter to compute the state of the robots and ball in the field. The following is an introduction to the linear filter, a simplified kind of Kalman filter [BRKN88].

1 Kalman Filter Loop

The Kalman filter loop is shown in Fig. F-1.

Fig. F-1. Linear filter model

2 Linear Filter Equations

The meaning of some symbols as follow:

-- y_n is the observed value $\dot{x}_{n,n}^* = \dot{x}_{n,n-1}^* + \frac{h_n}{T}(y_n - x_{n,n-1}^*)$

-- $x_{n,n-1}^*, \dot{x}_{n,n-1}^*$ is the estimated position and velocity value at time n based on the information before n, i.e. n-1, n-2, n-3, ...

-- $x_{n,n}^*, \dot{x}_{n,n}^*$ is the filtered estimation position and velocity value at the n based on the information before n+1, i.e. n, n-1, n-2, ...

(1) The position and velocity *g-h tracking update (filtering) equations*:

$$\dot{x}_{n,n}^* = \dot{x}_{n,n-1}^* + g_n (y_n - x_{n,n-1}^*)$$

$$\dot{x}_{n,n}^* = \dot{x}_{n,n-1}^* + \frac{h_n}{T} (y_n - x_{n,n-1}^*)$$

These equations provide an updated estimate of the present target velocity and position based on the present measurement of target range y_n as well as on prior measurements.

These equation are called *filtering equations*.

(2) The *transition equations* or *prediction equations*:

$$\dot{x}_{n+1,n}^* = \dot{x}_{n,n}^*$$

$$x_{n+1,n}^* = x_{n,n}^* + T\dot{x}_{n+1,n}^*$$

These equations allow us to transition from the velocity and position at time n to velocity and position at time $n+1$ and they all called the transition equations. We assume the estimated velocity at time $n+1$, $\dot{x}_{n+1,n}^*$, is equal to the value $\dot{x}_{n,n}^*$ at time n , because a constant-velocity target model is assumed.

(3) Combined the prediction equation with the filtered equation to yield the following *prediction update equation*:

$$\dot{x}_{n,n}^* = \dot{x}_{n,n-1}^* + \frac{h_n}{T} (y_n - x_{n,n-1}^*)$$

$$x_{n+1,n}^* = x_{n,n-1}^* + T\dot{x}_{n+1,n}^* + g_n (y_n - x_{n,n-1}^*)$$

Appendix F. Linear Filter

3 Stability of Constant g - h Filter

The **transition error** of the resulting signal is the sum of all the lag errors for a step change in velocity given by

$$D_{y_n} = \sum_{n=0}^{\infty} (y_n - x_n)^2$$

The total error is thus the error due to input noise and filter lag and given by the cost function

$$ERROR = VAR(y_n) + \lambda(D_{y_n})$$

where λ is a Lagrange multiplier that determines the importance of each portion of the error. The result filters gains are related by

$$h = \frac{g^2}{2 - g}$$

Stability is often a concern in designing a circuit for a filter. The literature show the constant g - h filter is stable when the filter parameters g and h obey the following conditions:

$$g > 0; h > 0; (4 - 2g - h) > 0$$

4 Relationship of Kalman Filter and Linear Filter

$$x_{n,n}^* = \left[\frac{x_{n,n-1}^*}{VAR(x_{n,n}^*)} + \frac{y_n}{VAR(Y_n)} \right] \frac{1}{1/VAR(x_{n,n}^*) + 1/VAR(Y_n)}$$

$$x_{n,n}^* = x_{n,n-1}^* + \frac{VAR(x_{n,n}^*)}{VAR(y_n)} (y_n - x_{n,n-1}^*)$$

$$x_{n,n}^* = x_{n,n-1}^* + g_n (y_n - x_{n,n-1}^*)$$

$$g_n = \frac{VAR(\dot{x}_{n,n}^*)}{VAR(y_n)}, \quad h_n = \frac{COV(\dot{x}_{n,n}^*, \dot{x}_{n,n}^*)}{VAR(y_n)}$$

5 Kalman Filter in Matrix Notation

$$X_n = \begin{bmatrix} x_n \\ \dot{x}_n \end{bmatrix} = \text{state vector,}$$

$$\Phi = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} = \text{state transition matrix for constant-velocity trajectory.}$$

$$U_n = \begin{bmatrix} 0 \\ u_n \end{bmatrix} = \text{dynamic model driving noise vector}$$

A random system dynamics model in matrix form $X_{n+1} = \Phi X_n + U_n$

A trivial measurements equation in matrix form. It is given by $Y_n = M X_n + N_n$, where

$$M = \begin{bmatrix} 1 & 0 \end{bmatrix} = \text{observation matrix,}$$

$$N_n = \begin{bmatrix} v_n \end{bmatrix} = \text{observation error,}$$

$$Y_n = \begin{bmatrix} y_n \end{bmatrix} = \text{measurement matrix}$$

Predictin equation:

$$X_{n+1,n}^* = \Phi X_{n,n}^*$$

Filtering equation:

$$X_{n,n}^* = X_{n,n-1}^* + H_n (Y_n - M X_{n,n-1}^*), \quad H_n = \begin{bmatrix} g_n \\ \frac{h_n}{T} \end{bmatrix}$$

Weitght equation:

$$H_n = S_{n,n-1}^* M^T [R_n + M S_{n,n-1}^* M^T]^{-1}$$

Appendix F. Linear Filter

Predictor covariance matrix equation:

$$S_{n,n-1}^* = COV(X_{n,n-1}^*), S_{n,n-1}^* = \Phi S_{n,n-1}^* \Phi^T + Q_n$$

Covariance of random system dynamics model noise vector U :

$$Q_n = COV(U_n) = E[U_n U_n^T]$$

Covariance of measurement vector $Y_n = X_n + N_n$:

$$R_n = COV(Y_n) = COV(N_n) = E[N_n N_n^T]$$

Correction equation (covariance of smoothed estimate):

$$S_{n-1,n-1}^* = COV(X_{n-1,n-1}^*) = (I - H_{n-1}M)S_{n-1,n-2}^*$$