8-27-2008

Control of Autonomous Robot Teams in Industrial Applications

Athanasios Tsalatsanis
University of South Florida

Control of Autonomous Robot Teams in Industrial Applications

by

Athanasios Tsalatsanis

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
Department of Industrial and Management Systems Engineering
College of Engineering
University of South Florida

Co-Major Professor: Ali Yalcin, Ph.D.
Co-Major Professor: Kimon Valavanis, Ph.D.
Tapas Das, Ph.D.
Sudeep Sarkar, Ph.D.
Nikos Tsourveloudis, Ph.D.

Date of Approval:
August 27, 2008

Keywords: mobile robot teams, autonomous systems, hybrid control, control architecture, supervisory control

Dedication

This work is dedicated to all those who made it possible. To my family for their love and support throughout these years. To my advisors, Dr. Valavanis and Dr. Yalcin, for their guidance, support and friendship that made me feel as a part of their families. I would like to thank the members of my dissertation committee Dr. Das, Dr. Sarkar, Dr. Tsourveloudis and Dr. Kaw for their time and comments on my work. I would also like to thank Dr. Kandel for his suggestions on my research. Finally, I would like to specifically thank my Laura for her love, support and inspiration.

Acknowledgments

This research was supported in part by two grants ARO W911NF-06-1-0069 and SPAWAR N00039-06-C-0062.

Table of Contents

List of Tables

List of Figures

Nomenclature

Table N.1 Variables used in Chapters 3 and 4

x, y	Coordinates of a pixel in image plane
RGB, YC_bC_r, HSI	Color spaces
T	A threshold value
H	Function for image's histogram
g, f	Grayscale and color image functions respectively
I_k	Variance of illumination in direction k
r	Correlation coefficient
d_x, d_y	Pixel disparities
Z	Depth between an object and the camera(s)
X	Distance between the object and the camera on the horizontal axis
B	Baseline
x_{pi}	Pixel coordinate on X axis
U_j	Range data from sensor j
x', y'	Vehicle's velocity and angular velocity respectively
φ	Angle between the camera and the X axis

Table N.2 Variables used in Chapter 5

$\mathbf{X}_k$	The state vector (system's model) at time k
$\mathbf{x}_k = \left[x_k \; y_k \; \vartheta_k\right]^T$	The vehicle's posture at time k
$\mathbf{u}_k = \left[V_k \; \Omega_k\right]^T$	Velocity vector consisting of the vehicle's linear and angular velocities at time k
$\mathbf{gps}_k = \left[gps_{x,k} \; gps_{y,k}\right]^T$	GPS readings related to vehicle's position at time k
$\mathbf{imu}_k = \left[ax_k \; eax_k \; ay_k \; eay_k \; angle_{z,k}\right]^T$	IMU readings related to the vehicle's accelerations, errors in acceleration, and steering angle at time k
$\mathbf{odo}_k = \left[odo_{x,k} \; odo_{y,k} \; odo_{\vartheta,k}\right]^T$	Odometer readings related to the vehicle's position at time k
$\mathbf{cam}_k^i = \left[dcam_{x,k}^i \; dcam_{y,k}^i\right]^T$	Distance between the i^{th} landmark and the vehicle as measured by the camera at time k
$\mathbf{las}_k^i = \left[dlas_{x,k}^i \; dlas_{y,k}^i\right]^T$	Distance between the i^{th} landmark and the vehicle as measured by the laser range finder at time k
$\mathbf{lan}_k^i = \left[lan_{x,k}^i \; lan_{y,k}^i\right]^T$	Position of the i^{th} landmark at time k
$\mathbf{d}_{j,k}^i = \left[dx_{j,k}^i \; dy_{j,k}^i\right]^T$	Distance between the i^{th} landmark and the vehicle as measured by the j^{th} range sensor at time k
$\mathbf{n}_k$	Zero mean Gaussian noise with covariance $\mathbf{Q}_k$
$\mathbf{w}_k$	Zero mean Gaussian noise with covariance $\mathbf{R}_k$
$\mathbf{Z}_k$	Measurement's model at time k
$\mathbf{P}_k$	Covariance matrix at time k

Table N.3 Variables used in Chapter 6

k	Number of task
j	Number of robot
Σ_j	Set of events that Robot j can execute
$\mathrm{Q_j}$	Set of states Robot j
δ_j	Transition function for Robot j
q_{j0}, Q_{jm}	Initial and final states Robot j
$G_j = (\Sigma_j, \mathrm{Q}_j, q_{j0}, \delta_j, \mathrm{Q}_{jm})$	The Finite Automaton representing the uncontrolled behavior of the Robot j
$R_n = (\Sigma_n, \mathrm{Q}_n, x_{n0}, \xi_j, \mathrm{X}_{nm})$	Requirements model
S	Specifications model
s	A string
$U(s)$	Utility of a string
$V(s)$	Maximum utility value for a sting
$U_N(s)$	Normalized utility function for a string

Control of Autonomous Robot Teams in Industrial Applications

Athanasios Tsalatsanis

ABSTRACT

The use of teams of coordinated mobile robots in industrial settings such as underground mining, toxic waste cleanup and material storage and handling, is a viable and reliable approach to solving such problems that require or involve automation. In this thesis, abilities a team of mobile robots should demonstrate in order to successfully perform a mission in industrial settings are identified as a set of functional components. These components are related to navigation and obstacle avoidance, localization, task achieving behaviors and mission planning. The thesis focuses on designing and developing functional components applicable to diverse missions involving teams of mobile robots; in detail, the following are presented:

1. A navigation and obstacle avoidance technique to safely navigate the robot in an unknown environment. The technique relies on information retrieved by the robot's vision system and sonar sensors to identify and avoid surrounding obstacles.
2. A localization method based on Kalman filtering and Fuzzy logic to estimate the robot's position. The method uses information derived by multiple robot sensors such as vision system, odometer, laser range finder, GPS and IMU.

3. A target tracking and collision avoidance technique based on information derived by a vision system and a laser range finder. The technique is applicable in scenarios where an intruder is identified in the patrolling area.
4. A limited lookahead control methodology responsible for mission planning. The methodology is based on supervisory control theory and it is responsible for task allocation between the robots of the team. The control methodology considers situations where a robot may fail during operation.

The performance of each functional component has been verified through extensive experimentation in indoor and outdoor environments. As a case study, a warehouse patrolling application is considered to demonstrate the effectiveness of the mission planning component.

Chapter 1

Introduction

1.1 Robots in Industry

The use of robotic based solutions in industry is driven by the need for safety, quality and efficiency. The increasing demand for inexpensive products of higher quality and increased reliability requires frequent modifications to the production process. The flexibility of the system to adapt to new modifications is bounded by the overpriced and time consuming training of specialized personnel. In addition, many activities in industrial applications such as manufacturing, underground mining, toxic waste cleanup and material storage/ handling of chemicals, take place in hazardous environments harmful to human health. To address the issues of safety, quality and efficiency in industry it is essential to reduce the human presence from particular applications and assign them to autonomous robotic based solutions.

The scope of this chapter is to familiarize the reader with terms such as robots, robot teams, control architectures and functional requirements. Brief examples of different robots and control architectures are presented.

1.2 Robots and Robot Teams: Definitions

The term robot is derived from the Czech word robota (force labor) and it was first used in the movie Rossum's Universal Robots by Karel Capek. In this movie small, artificial, humanoid beings named robotniks were obeying their lord's commands. However, the world of robotics have come a long way since robotniks. Different types of autonomous robots such as robotic manipulators, mobile robots, aerial and underwater vehicles have been developed for industrial, military and civilian applications. The term autonomous refers to systems that are capable of performing a number of operations for long periods of time without any external interference [1]. The robotic manipulators shown in Fig. 1.1a, are articulate devices of finite size able to perform high reparative operations such as pick and place activities, car body painting, welding and material handling. These robots are used in automotive and electronics industry and they are composed of two parts: a stationary part fixed on the ground or on a platform and a moving part able to move with 2 to 6 degrees of freedom. The robotic manipulators have limited range of operation because of their stationary part. Alternatively, robots with mobility capabilities have been developed. Depending on the operational environment, these robots are classified into three main categories: Unmanned ground (UGVs), aerial (UAVs) and underwater (UUVs) vehicles. These robots consist of a set of sensors and actuators onboard a moving platform. The moving platform for the case of UGVs or mobile robots, shown in Fig. 1.1b, is a car like vehicle with two (2) or more wheels. In the case of UAVs, see Fig. 1.1c, the moving platform has the form of an airplane or a rotorcraft (helicopter) and in the case of UUVs, see Fig. 1.1d, the form of a miniature size subma-

rine. Robots with mobility capabilities have been extensively used in military applications such as battlefield surveillance and target tracking [2] or humanitarian minefield demolition [3], [4], while lately they become available for civilian applications such as search and rescue [5], [6] and automated traffic monitoring [7].

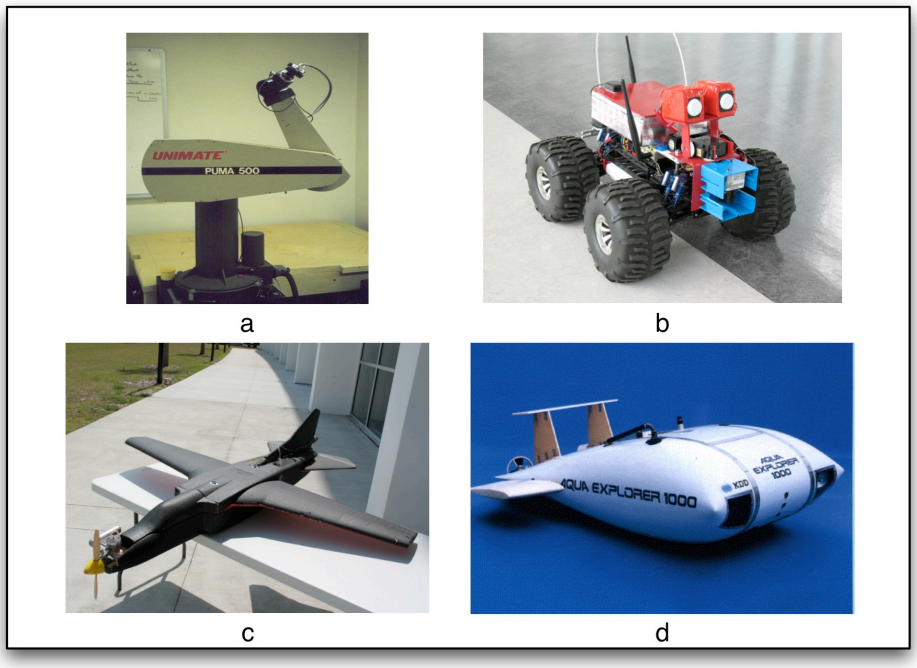


Fig.1.1 The Puma 500 robotic manipulator (a); a mobile robot (b) and an aerial vehicle (c) from the USF Unmanned Systems Laboratory; the underwater vehicle Aqua Explorer 1000 by KDD Co. (d)

Several well structured applications such as “pick and place” or material handling activities can be performed by single robotic solutions. However, as the complexity and the requirements of the application increases, the use of multi robotic solutions is more suitable. For instance, using a single robot in applications such as patrolling or mapping

is a viable solution but not an effective one. For example, for a warehouse patrolling mission a mobile robot should carry the appropriate sensors to identify fire, chemical leaks and unauthorized human presence. Installing all these sensors on a single robot is not an optimal decision since the robot could be trapped in a fire resulting in total loss of these sensors. In addition, the probability of robot failure increases as the number of sensors increases disqualifying the robot from achieving its mission. Furthermore using one robot instead of many significantly increases the time required to accomplish the mission. A multi robotic system would be more efficient for such applications. The overall mission would be decomposed into tasks allocated to the robots according to the set of sensors each robot carries. Individual robot failures would be addressed by assigning the tasks of the failed unit to one or more other robots. In this manner, the probability of the multi robotic system to achieve the overall objective increases even though some of the system's members are inoperative.

Multi robotic systems can be classified into three main categories: robot teams, cooperative robot teams and swarms. The term *robot teams* refers to small size group of robots with different properties or capabilities. For example, a robot team can have members from aerial, underwater and ground vehicles, all working together to achieve a set of common objectives. The overall objective is decomposed into tasks for each robot. Cooperation between the robots of the team to perform a task is not allowed. On the other hand the cooperative robot teams consider combined effort on task execution allowing two or more robots to work on the same task supplementing each other. Finally, the term swarms refers to large size groups of robots with identical properties and capabilities. In

this work a team of robots consisting of mobile robots equipped with overlapping sensory capabilities, to allow multiple selections of robots for a task is considered.

1.3 Functional Requirements

All mobile robot teams operate under a set of functional requirements related to the team's ability to perform a mission. These requirements can be broken down into seven main components:

1. Mission planning
2. Communication
3. Task execution
4. Motion control
5. Localization
6. Navigation and obstacle avoidance
7. Task achieving behaviors

Fig. 1.2 provides a schematic representation of the functional components and their relationship.

The Mission Planning component includes all the features related to higher level control, such as mission decomposition, task allocation, optimization, robot cooperation and observation. It receives information through the communication component from all

the robots in the team regarding their posture, functionality and task status. The Mission Planning component generates the task assignments for each robot.

The Task Execution component is responsible for mission planning at the robot's level. It coordinates all the robot's actions to achieve a set of tasks utilizing instructions on how to perform the current task from the Task Achieving Behaviors component. The Task Execution components notifies the Mission Planner with the status of the robot (operational or non operational) and of the task (completed or uncompleted).

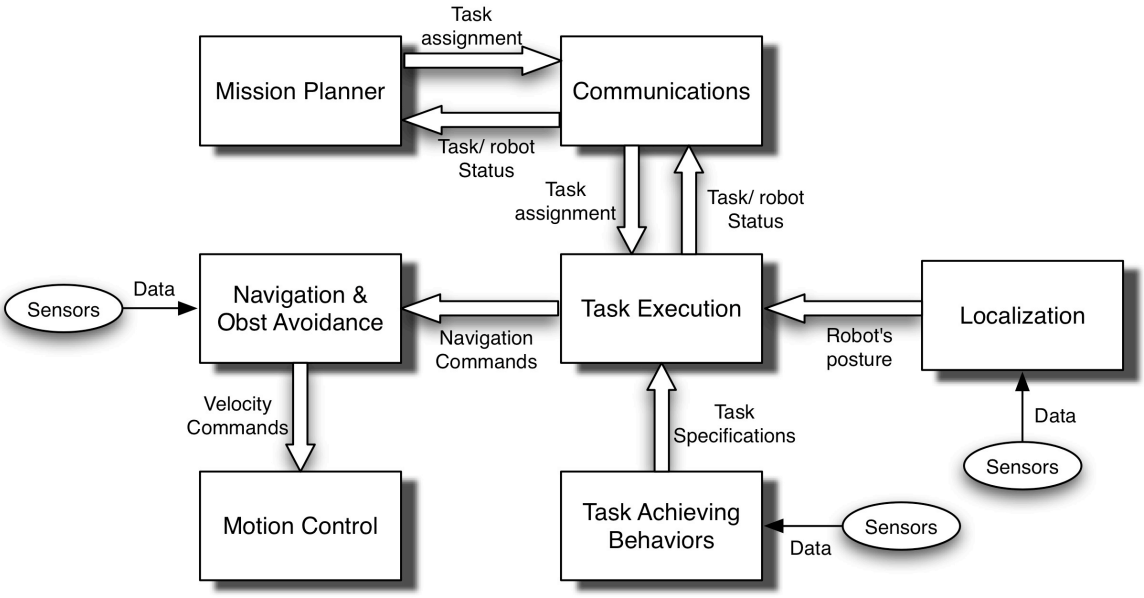


Fig. 1.2 Functional components for a team of mobile robots

The Task Achieving Behaviors component is a modular depository of behaviors that can be activated individually or in parallel related to task specifications.

The Motion Control component consists of all the physical mechanisms of the robot such as actuators and wheels responsible for the robot's mobility.

The Localization Component provides estimates of the robot posture, velocity, acceleration and orientation. It receives information from a set of position sensors such as GPS, IMU and odometer, and reports to the Task Execution component the current posture of the robot.

Navigation and Obstacle Avoidance refers to the set of algorithms responsible for the safe passage of the robot from an initial point to a final destination. The navigation component receives inputs from the Task Execution component and the robot's sensors and provides velocity commands to the Motion Control component.

The afore-described functional components are common to all mobile robot teams regardless of the mission's nature. A team of mobile robots has requirements for mission control, task execution, motion control, localization, communication and navigation whether it is assigned with a planetary exploration or with a toxic waste cleanup mission. However, there is significant diversity on how all these functional components interact with each other resulting in various degrees of flexibility for the robot team. This interaction between the functional components is represented by control architectures.

1.4 Control Architectures

In robot teams, there are numerous approaches for control architectures ranging from fully centralized to fully distributed approaches. The centralized control architec-

tures, shown Fig. 1.3, use one robot or a central controller as the team's leader. The controller is responsible for the team's actions, from mission planning to obstacle avoidance. Primary advantage of the centralized control architectures is that the central controller has exact knowledge of each robot's activities, which allows for optimal planning. However, centralized approaches present many disadvantages. Since the central controller is responsible for all the actions in the team, the computational complexity of the system increases as the number of robots increases, which implies that the response of the system in a dynamic environment is rather slow. Furthermore, every team member communicates directly with the central controller increasing the communications bandwidth. Finally, if the communications link or the central controller fails, the team's members become vulnerable to dangerous citations. Fully centralized approaches are suited for applications as Robocup [8] with small teams of robots operating in static environments.

The distributed control architectures shown in Fig. 1.4, allocate the team's intelligence to the robots. Each robot operates independently and has the capability to determine its actions based on the requirements of the mission and the actions of the other robots. Fully distributed systems are easier to implement but have significant drawbacks usually in the area of optimization since there is no overall monitoring of the system's processes. Additional functional requirements arise to resolve situations when more than one robots attempt to execute the same task (conflicts). Fully distributed control architectures have been used for hazardous waste cleanup [9] and humanitarian de-mining [10].

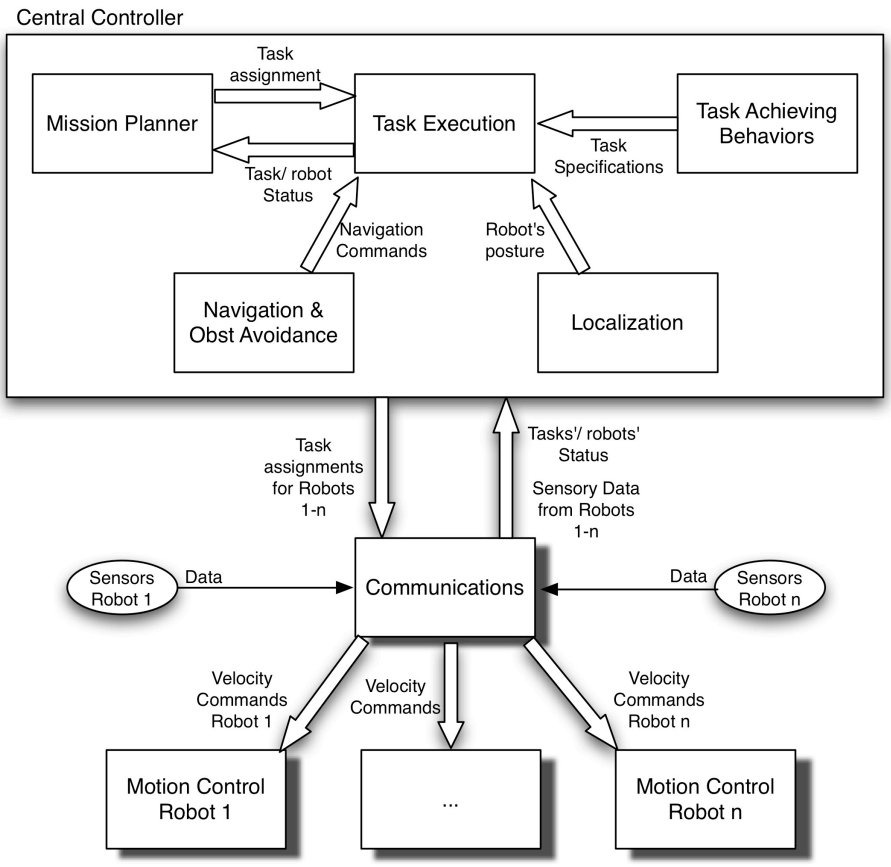


Fig. 1.3 Centralized control architecture for a team of mobile robots

Hybrid control architectures, shown in Fig. 1.5, combine characteristics from both centralized and distributed approaches. The intelligence of the robot team is separated into layers of control. The higher layer is responsible for the overall performance of the team while the lower layers are responsible for each robot's functionalities such as task execution, navigation and localization. The control design allows the robots to carry on with their task assignments even if the communications with the mission planner are lost. In addition, communications between the mission planner and the team's members are limited to task assignments, task completion and operational status, which implies that hybrid control architectures do not require broad communication bandwidth.

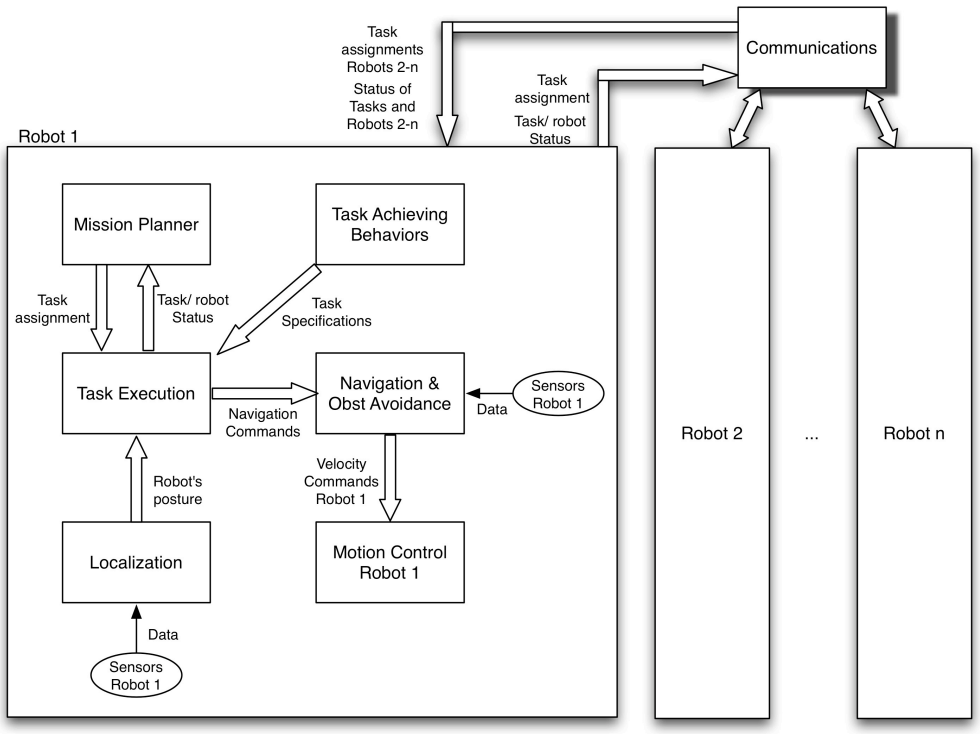


Fig 1.4 Distributed control architecture for mobile robot teams

1.5 Functional Components

In Fig. 1.3-1.5, the team's functional components appear as structural modules in each control architecture. As noted earlier, these functional components are common to all missions involving mobile robot teams. However, the functional component design could be either generic, applicable to any mission, or mission specific. For example, the Mission Planner of a space exploration mission cannot be used in a material storage application. Nevertheless, both Mission Planners share common characteristics such as how to decompose the overall mission into tasks for each robot or how to handle resource

failures and repairs. On the other hand, the navigation and obstacle avoidance component would be the same in both missions, space exploration and storage handling, assuming that the same sensors are used.

The design of the functional components is a challenging problem. Each component should demonstrate characteristics that will allow the robot team to accomplish its mission effectively.

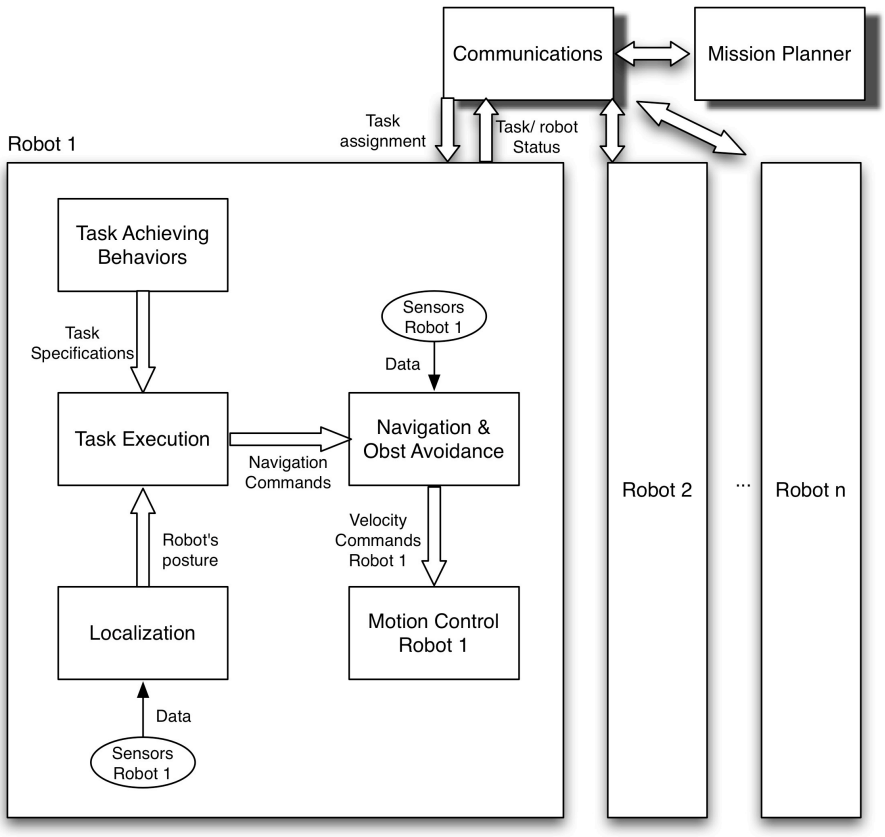


Fig. 1.5 Hybrid control architecture for mobile robot teams

The main issues that should be addressed in the design of the Mission Planner are:

1. Mission decomposition and task allocation. Each robot of the team carries a set of sensors appropriate for a set of tasks. Based on these sensor capabilities, the Mission Planner identifies a suitable robot for each task.
2. Control Optimization. Various optimization criteria such as shortest route or use of the low-priced robot can be utilized to maximize the performance of the robot team and at the same time minimize the associated cost.
3. Robot failures and repairs. A very important consideration in the Mission Planner design is that of resource failures and repairs. A failed robot could jeopardize the team's mission. The Mission Planner must demonstrate the ability to respond to robot failures by reallocating tasks to the remaining robots of the team. At the same time, since a repaired robot is a significant asset to the team's performance, the Mission Planner must be able to reintegrate the repaired robot into the team.
4. Completion of the overall objective. All tasks related to the mission must be completed under the assumption that there are available resources.
5. Deadlocks and conflicts. The design of the Mission Planner must avoid situations where more than one robot are assigned to the same task or a robot reaches a state from which it cannot get out of.
6. Computational requirements. The Mission Planner must be able to adapt to the changes in the team's environment in real time.

In the design of the Navigation and Obstacle Avoidance component there are two key issues:

1. Velocity commands. Based on the current position of the robot and a final destination, a set of algorithms computes the appropriate velocity commands that will move the robot to its target.
2. Obstacle Avoidance. Each robot carries range sensors that provide information about the surrounding environment. The Navigation and Obstacle Avoidance component utilizes the sensor readings to identify a safe, obstacle free passage for the robot.

The Localization component must be able to utilize readings from position and range sensors to determine the posture of each robot. The Mission Planner requires this information for task allocation. Also, the Navigation and Obstacle Avoidance component uses the posture information to derive the velocity vectors for the robot's motion.

Finally, the Task Achieving Behaviors component encapsulates a set of behaviors each robot should demonstrate to achieve the mission. Since not all the robots are able to perform all the tasks in a mission, the design of the Task Achieving Behaviors component should be modular in a way that each robot can activate the appropriate behavior to perform a given task.

1.6 Patrolling and Inspection

An effective functional components design can be demonstrated in various industrial applications that utilize mobile robot teams. Such application is that of autonomous patrolling and inspection where a team of mobile robots is assigned to guard and

monitor an industrial area (i.e. warehouse). In general patrolling refers to a group of individuals that perform security functions, collect information and report unexpected activities. Characteristics that a patrolling group should demonstrate are:

1. Mission decomposition. Each member of the patrol team is assigned to a region of the patrolled area. Not all regions have the same requirements for patrolling, thus each of the team's members can be assigned to a specific region.
2. Optimization. Depending on the number of members in the group, a minimization on the number of regions and the member routes can be achieved to completely cover the patrolled area.
3. Patrol frequency. Each region of the patrolled area must be revisited in regular or in random time intervals.

These characteristics can be incorporated into the design of the functional components and enable a team of mobile robots to perform patrolling missions.

1.7 Motivation

From higher level functions such as mission decomposition and task allocation to lower level operations such as collision avoidance, the design of each functional component is vital to the successful operation of the robot team. A robot that cannot navigate safely cannot perform task requiring robot movement, and a team of robots cannot function efficiently unless there is a way of task allocation.

In literature, there is significant work in the design of each individual functional component. This work investigates how these functional components can interact with each other and allow the robot team to effectively perform a mission. Consider the situation where the Navigation and Obstacle Avoidance component uses information from a laser range finder to avoid collision with an obstacle, while the Localization component uses the same sensor to estimate the position of the robot causing a conflict in sensor utilization. The design of each functional component has to consider the interaction between the components and resolve conflicts in sensor utilization.

1.8 Research Objectives

The goal of this research is to design functional components common to a wide range of autonomous mobile robot teams such as Mission Planning, Navigation and Obstacle Avoidance, Localization and Task Achieving Behaviors and investigate their deployment under the umbrella of a hybrid control architecture. The specific objectives are:

1. Identify functional components applicable to any mission involving mobile robot teams.
2. Design individual functional components applicable to a warehouse patrolling mission
3. Demonstrate individual functional component design and interaction using an autonomous patrolling application

1.9 Research Contributions

The main contribution of this research is that it integrates all the key elements required for the operation of a robot team as a whole. Through this research, functional components essential for most robot teams are identified and methodologies are developed to allow a robot team to complete missions such as warehouse patrolling.

The rest of this section summarizes the contributions related to component design methodologies and the warehouse patrolling application.

A novel depth estimation technique is introduced in the design of the Navigation and Obstacle Avoidance component. The technique uses information from two cameras to derive depth estimates between the robot and an object. The novelty of this technique is that instead of calibrating the cameras and using standard stereo vision equations to derive depth, parameters such as the image size, the angle of view of the cameras and the relative position of two different captures of the same scene have been used to derive a depth estimation. The developed technique presented maximum error of 8% in depth measurements up to 8m, better than most approaches in literature.

In the design of the Task Achieving Behaviors, a vision based target tracking technique is presented. A main contribution of this technique is that it uses a stereo vision system where both cameras track a target independently, providing a redundant mechanism that helps avoiding loosing the target. This means that even if one camera ‘loses the target’, it can retrieve information from the other camera to find it again. In addition the robot’s direction is controlled by the pan/tilt angles of the cameras, allowing the robot to

avoid obstacles and keep tracking a target, or to keep tracking a target that moves on uneven terrains.

For the Localization component, a fuzzy Extended Kalman Filter methodology is developed that fuses information from multiple sensors to derive an estimate of the robots posture. Main contribution of this work is the consideration of five distinct sensors: GPS, IMU, stereo vision system, laser range finder and odometer, while most related approaches do not exceed three sensors. In addition, the Fuzzy Logic controller design allows for incorporation of the error in sensor readings into the EKF without the use of an error model. The error in sensor readings is approximated by multiple zero mean Gaussian distributions and it is included in the covariance matrix of the measurement model.

Finally, for the Mission Planning component a limited lookahead control policy is presented. Contributions of this work focus in the areas of computational complexity and of task allocation. In this work, instead of constructing the complete supervisor, a limited lookahead policy is utilized that constructs a lookahead supervisor based on the evolution of the system in real time. The proposed methodology offers an advantage to highly dynamic systems, since the control model can change on the fly. In addition, the design of the Mission Planning component contributes to the warehouse patrolling application by modelling the mission using Discrete Event Systems.

1.10 Outline

This thesis is organized as follows. Chapter 2 is dedicated to literature review. A number of well established architectures in teams of mobile robots are presented. The chapter focuses on centralized, distributed and hybrid control approaches.

Chapter 3 presents a Navigation and Obstacle Avoidance method for a mobile robot. The method uses range information from a stereo vision system and a set of sonar sensors to safely navigate the robot in an indoor environment. Also, this chapter presents a simple method of computing range from a stereo vision system.

Chapter 4 discusses the Task Achieving Behavior functional component. Patrolling applications require the ability of recognizing and tracking unauthorized presence in a secure facility. A vision based target tracking method is presented that utilizes a stereo vision system and a laser range scanner to enable a mobile robot to track an intruder and notify the authorities.

In Chapter 5 a fuzzy Extended Kalman Filter is presented that uses information from multiple sensors: GPS, IMU, stereo vision system, laser range finder and odometer to determine the posture of a vehicle travelling in indoor and/or outdoor environments.

Chapter 6 is devoted to the Mission Planner component. A dynamic task allocation and controller design methodology for cooperative robot teams based on limited lookahead policies is presented.

Finally, Chapter 7 presents the contributions of the completed work and discusses directions in future work.

Chapter 2

Literature Review

2.1 Introduction

Control architectures for the control of a single autonomous mobile robots are designed based on three main paradigms: deliberative, reactive and hybrid deliberative/reactive [1]. Main characteristic of the deliberative architectures, as in [11], [12], [13], is their hierarchical structure. Each layer of the architecture is dedicated to a different function of the robot (i.e. sensing, thinking and acting). In this sense, robot's actions are determined by reasoning rather than just reaction to sensory information. On the other hand, reactive architectures, as in [14], [15], [16], consist of a set of behavioral modules each designed for a different function of the robot. All the modules are activated in parallel and respond to direct sensory information. Examples of these behavioral modules are: obstacle avoidance, map building and exploration. Finally, hybrid deliberative/ reactive architectures, as in [17], [18], [19], [20], incorporate the reasoning part of the deliberative architectures and the behavioral modules of the reactive architectures. An additional planning module is used to provide reasoning on the activation of the appropriate behavior according to the sensory information and to the robot's mission.

These paradigms are used to design architectures for single mobile robot control. When the number of robots increases and mobile robot teams are shaped, architectures that will coordinate the robots' efforts to achieve a common mission are required. These architectures are classified into three main categories: Centralized, Distributed and Hybrid architectures. In the centralized approaches, a central controller is responsible for all the functions of the team. Each robot has no intelligence and all sensor readings are transmitted to the central controller. The central controller decides for every action that each robot should take, for example, which task to perform and how to avoid an obstacle. On the other hand, distributed architectures distribute the intelligence among the robots. Every robot has modules for behaviors such as obstacle avoidance and task execution. The robot team performs task allocation collectively. Finally Hybrid control architectures distribute a layer of control to the robots and keep the task planning and task allocation to a central computer. This central computer can be a robot of the team. The rest of this chapter summarizes well established architectures for mobile robot teams.

2.2 Distributed Robot Architectures

2.2.1 ALLIANCE

ALLIANCE [9] is a distributed software architecture build for heterogeneous mobile robot control emphasizing in fault tolerance capabilities. It has been designed for small teams of robots that have to perform independent tasks. All the robots have the

capability to determine their own actions based on the requirements of the mission, the activities of other robots, the current environmental conditions and the robot's internal state.

The architecture uses a hierarchical behavioral based model. The lower level behaviors correspond to primitive survival behaviors such as obstacle avoidance, while the higher-level behaviors correspond to task achieving functions such as map building or exploring.

ALLIANCE incorporates the use of mathematically modelled motivations such as impatience and acquiescence. The robot impatience incorporates two parameters: The time in which one robot is willing to allow another robot to affect the motivation of a behavior set and the time that a robot allows another robot to accomplish its task before becoming impatient. The robot acquiescence determines the time that a robot needs to finish its task and assigns the task to another robot it that time is exceeded, and the time before assigning the robot to another behavior.

Significant contributions of the ALLIANCE architecture are the distributed fault tolerance and adaptivity. Fault tolerance is the ability of the robot team to respond to individual robot failures or failures in communications that may occur during the mission. The author proposes the dynamic re-selection as response to fault tolerant. Adaptivity is the ability of the robot team to change its behavior over time to response to a dynamic environment or changes to the team's mission.

The author demonstrates the effectiveness of the architecture using a team of three robots performing a laboratory version of hazardous waste cleanup.

2.2.2 DRS

DRS [21] addresses the issue of dynamic distributed task allocation when more than one robot can perform the same task. The architecture is fully distributed and assumes no centralized mechanism such as CPU or shared memory. Further the communications between the robot are considered limited. Each robot makes decisions independently and all robots cooperate to achieve a common goal. The architecture has been designed for applications in space exploration, defence operations and other automation systems where high reliability is required.

DRS is based on a set of Distributed Mutual Exclusion algorithms that use a "sign-board" for inter-robot communications. Each robot can read and write on the board and any other robot of the team can read the message. The characteristics of the Distributed Mutual Exclusion algorithms are:

1. Mutual exclusion: The capacity of a resource should never be exceeded
2. Deadlock free: The algorithm should not result in situations where none of the robots can ever get a resource
3. Lockout free: the algorithm should ensure that a robot requested access to a resource eventually it will have its turn to check the resource out.

One of the issues that it is not discussed in DRS is that of dynamic reallocation due to robot failures.

2.2.3 ACTRESS

ACTRESS [22] is a decentralized control system for multi-robot systems that addresses the issues of communication, task assignment and path planning among heterogeneous robotic agents. Each robot is considered to have the ability to understand the target of tasks, recognize the surrounding environment, act and manage its own conditions. Further, each robot has the ability to communicate with any other robot of the team to work in parallel for specific tasks or avoid interference for other tasks.

The contribution of ACTRESS is the ability of recruitment. Many robots can work on a task in parallel and request help when needed. The efficiency of ACTRESS has been tested on an application where mobile robots perform a box-pushing task.

2.2.4 CAMPOUT

CAMPOUT [23] is a distributed control architecture based on multi-agent behavior-based methodology. The architecture consists of a set of elementary architectural mechanisms for behavior representation, behavior composition and behavior coordination, group coordination and the interface between them.

The behavior representation is implemented using finite state machines. A behavior is formalized as a mapping that relates each possible sensory information sequence to an action. The most desired actions are assigned the value 1 and the undesired the value 0.

Behavior composition refers to the mechanism used to build high-level behaviors by combining lower level behaviors based on the behavior coordination mechanisms.

Behavior Coordination: Behavior coordination mechanisms are divided into two complementary classes: arbitration and command fusion.

Arbitration mechanisms select one behavior from a group of competing behaviors and give the complete control of the system until the next cycle. CAMPOUT uses a priority-based arbitration where behaviors with high priority are allowed to suppress the output of behaviors with lower priorities, and state based arbitration, which is based on DES.

Command Fusion mechanisms combine multiple behaviors to form a control action. This approach allows all the behaviors to contribute to the control of the system in a cooperative manner. CAMPOUT uses the following command fusion techniques:

1. Voting techniques that interpret the output of each behavior as votes for or against all possible actions
2. Fuzzy commands that use fuzzy inference to formalize the action selection process
3. Multiple behavior fusion mechanisms that select the best trade-off between the task objectives that satisfies the behavioral objectives

Group coordination is treated as the coordination of multiple distributed behaviors, where more than one decision makers are present. Finally, the robots communicate either by interaction or through direct communication.

The architecture is designed for planetary surface exploration. The authors of the paper do not present a failure recovery mechanism incorporate into the architecture.

2.3 Centralized Architectures

Centralized approaches for control architectures are used primarily in RoboCup (Soccer with robots), where a central computer receives inputs from the robots of the team and external sensors (vision system mounted on top of the soccer field) and decides on the strategy that the robots should follow.

2.3.1 CMUnited

CMUnited [8] is an architecture that perceives the overall system as a combination of robots, external sensors and a centralized interface computer. The complete system is fully autonomous. The external sensor (vision) identifies the positions of each robot and the ball and transmits this information to the centralized computer. The computer uses runs a set of different algorithms that evaluate the team's state and decide on what the robots should do next. The central computer sends command actions to the robots of the team.

This is a typical centralized control architecture. The major disadvantage of such architectures is that in case of communication failures each member of the team stops reacting.

2.4 Hybrid Architectures

2.4.1 GOFER

GOFER [24] is a sense-model-plan-act architecture designed to control a team of multiple robots in indoor automation applications. The architecture includes a task planning system, a task allocation system, a motion planner and an execution system. The task planner derives plans made of a high level action such as “go to position A” or “get object K”. The task allocation system orders and allocates tasks or actions to the robots. The motion planning system converts the high level actions into motion commands and finally the execution system monitors the execution and reacts to unexpected events.

The task allocation system is partially centralized. Each robot communicates with the task allocator to determine its current task. The task allocator receives orders from the task allocation system and generates plan structures and provides the available robots with a description of these plans. Each robot acquires a goal in two ways. The first is to ask or to be asked to perform a task and the second is to autonomously generate tasks with respect to the current situation. For example a robot may ask the other robots to determine what it can do for them. GOFER uses hierarchical Petri Nets for interpretation of the plan decomposition and execution monitoring.

2.4.2 3TEAR

3T [25] is a three-layered architecture, with skills, sequencing and planning layers. The planner constructs partially ordered plans, listing tasks for the robot to perform according to some goals. The Sequencing layer decomposes the tasks that have been constructed by the planner to sets of actions. Each action corresponds to a set of skills that are activated in the Skills layer. Additionally, event monitors are activated in the Skills layer, notify the sequencing layer of completion of the set of actions.

2.4.3 A Hybrid Control Architecture for Mobile Robots

The architecture presented in [26] combines aspects of classic control and behavior-based control. Both continuous and discrete event systems are considered. The DES part is modelled using Petri nets. The control architecture consists of 4 layers: User interface, Action planning, Motion planning control and Obstacle avoidance. The higher level is the user interface that stores the instructions that a robot should follow. These instructions are a set of actions that the robot should perform according to the sensor readings. The Action planning layer considers the directions of the User interface and issues a reference for the Motion control level. The Action planning level is designed using Petri nets. The motion control level has a reactive control loop to avoid obstacles.

This architecture is designed to control a single robot. The concept of control that the authors propose can easily transferred in teams of robots, where the user interface and

action planning layers are located to a centralized computer and the layers of Motion planning and Obstacle avoidance are located in each robot.

2.4.4 Hybrid Algorithms of Multi-Agent Control of Mobile Robots

Authors of [27] combine methods from AI and neural network control to solve the problem of multi-agent robotic systems. The proposed architecture consists of a strategic (supervisor) and a tactical (local) level of control. The strategic level uses AI techniques to address task decomposition, optimal task allocation, global environment modelling and avoidance of collision between the robots. The tactical level controls actions such as obstacle avoidance, optimal path planning and control of robot actuators. Each robot is a complete independent system that is able to perform navigation, acquire sensory data and communicate with other robots and the supervisor.

One drawback of the paper is that the authors use external databases to store the complete paths of each robot. In this way the communication channels are always busy carrying information from the robots to the supervisor and to the database. It would be sufficient to store only the current positions of the robots in the central computer.

2.5 Hybrid Control Architecture for Autonomous Patrolling

The control architectures presented in this chapter are applicable to mobile robot teams. Applications such as autonomous patrolling can be modelled using any of these

architectures. The approach followed in this work is closer to hybrid control architectures. Each robot has a certain level of autonomy, which allows task execution while a central computer or a robot is responsible for mission planning. The control architecture is decomposed into three levels of control. The higher level is responsible for the task allocation between the robots, the middle level is responsible for the task execution (monitoring) while the lower level is responsible for each robot's operation. The overall architecture encompasses a continuous and a discrete control system. The continuous control system consists of a set of functional components that can be activated individually or in parallel and are related to each robot's task assignment. The discrete control system, Mission Planner, is a Discrete Event Dynamic Systems (DEDS) supervisor that allocates tasks to the robots and oversees the behavior of the robot team.

Chapter 3

Navigation and Obstacle Avoidance

3.1 Introduction

A novel, efficient and robust approach is presented for vision based depth estimation. The method does not require any camera calibration technique that adds computational load to the system, but it is based on the image size, the field of view of the camera(s) and the relative position of two image frames of the same scene. The image size is set by the user, the field of view is known by the camera's manufacturer, while the relative distance can be measured. No additional information is required for a given application.

A case study demonstrates the efficiency of the presented method for indoor mobile robot motion planning and collision avoidance. Simultaneous ultrasonic sensor and camera range measurements are used to identify obstacles and navigate the robot around them.

Ultrasonic sensors have the advantage of fast and accurate range measurements, but in certain cases, small surface objects, or objects situated at a wide angle related to the ultrasonic sensor(s), cannot be detected. Given these limitations, a vision system may

be used to identify obstacles within the field of view of the robot; in this chapter obstacles located at distances up to *8m* are identified accurately. This is achieved using a color space transformation, adaptive thresholding and the proposed method.

The YC_bC_r color space has been used to retrieve the obstacle's color from the acquired image; this choice over the HSI and CMYK color spaces is justified because the YC_bC_r color space demonstrated better performance in locating the object in question (however, results using all three color spaces are included for comparison purposes).

Both sonar sensors and cameras are activated and operate simultaneously and in parallel to obtain range measurements from common search areas in the front of the mobile robot. The algorithm allows for "back and forth" sonar / camera based obstacle identification, in the sense that although sonar sensors may initially identify a potential obstacle, camera data may be used to complement and decide about the presence / absence of potential obstacles and navigate around them. The implemented vision system consists of two uncalibrated rotated color cameras mounted on top of the robot at a distance of *45* cm from each other. Experimental results and comparisons using a parallel stereoscopic, rotated and monocular vision system are presented.

The computational complexity of the camera based range measurement is of order $O(n^2)$, where n is the dimension of a square image, while the complexity of the sonar based range measurement is of order $O(n)$, where n is the number of the sonar sensors.

Experimental validation and verification of the proposed method has been demonstrated using the ATRV-mini skid steering differential drive robot, equipped with *350 MHz PIII* processor and *256 MB* of RAM. The ATRV-mini uses *24* ultrasonic

sensors, a GPS, a compass unit and a vision system that consists of two pan/tilt video cameras, SONY EVI-D30. The robot runs *RedHat Linux 6.0* and *Mobility* interface. Obstacles are assumed to be of yellow, red, green, blue, or mixed color. No enhancements or modifications to the original system have been performed. Movement from an initial to a final goal point follows the fuzzy controller framework reported in [28].

Experiments confirm the effectiveness of the presented approach; the maximum computational error as well as the *Normalized Root Mean Square Error, rmse*, of range measurements using either a rotated, parallel, or monocular vision system is about *3.5% (rmse 0.023), 4.21% (rmse 0.025) and 2% (rmse 0.011)* respectively, for obstacles lying at a distance of *27–800 cm* from the robot. This computational error is considerably better compared to results presented in [29], [30], [31], [32], [33], [34], [35].

The accuracy and error margin related to sonar sensor range measurements depends heavily on the type of sonars used; however, in all cases, accurate data is returned within a *2 meter* distance.

The chapter is organized as follows. The next section refers to related work and comparisons. Section 3.3 describes the vision system algorithm, and Section 3.4 presents the motion algorithm. Section 3.5 is dedicated to experimental results. Section 3.6 discuses the case of monocular vision system (basically for completeness purposes) and Section 3.7 concludes the chapter.

3.2 Related Work and Comparisons

In general, vision based range measurements require knowledge of the camera intrinsic and extrinsic parameters. A widely used technique to acquire this knowledge is camera calibration where a set of linear equations with 12 unknown parameters needs be solved [36]. This technique is applicable to known or unknown scenes assuming that the intrinsic parameters of the camera do not change for different views. This is a valid assumption when pinhole cameras are used. The modern CCD cameras tend to automatically adjust their intrinsic parameters in order to acquire clear images. Thus, calibration has to be repeated every time the scene or the camera orientation changes.

Reported research in [29] and [30], estimates depth by fusing defocus and stereo with reported *rmse* errors of about *0.12* and *0.34*, respectively. In [31], a sub pixel stereo method is proposed with reported computational errors of more than *10%*. In [32], dynamic stereo images are used with reported computational error less than *5%*. In [33], a defocus method is used with errors of less than *5%*. In [34], the robustness of image based visual servoing is explored with reported errors of *6%*. Research in [35] combines a monocular CCD camera and an inertial sensor with an error of about *5.42%*. Reported research in [35] includes only simulation results, therefore, no comparison is made.

Regarding depth estimation, the presented approach differs from related work presented in [37] in which range through visual looming is computed using a monocular vision system. A major disadvantage of this approach is that an object's projection must fit entirely within the focal plane, as opposed to our approach where only a part of the

object is required for the same computation. Also the distance between the camera and the object is assumed to be measured from the object's center. In our approach, a distance value is calculated for each of the object's corners providing more accurate range measurement especially when the object is not parallel with the focal plane. Furthermore, even no calibration is required in [37] one has to know the physical size of the image plane as opposed to our approach where no such information is required.

The approaches presented in [38], [39], [40], [41], [42] do not use the vision system for range measurements. The research in [38] uses vision system information for robot localization and detection of static obstacles through an environment model, and sonar range measurements to detect moving obstacles. The approach followed in [39] for indoor navigation of robot teams uses sonar information to detect walls, open doors and corners through characteristic sonar footprints, while the vision system is activated to describe what has already been detected. Similarly, the work in [40] uses range information from ultrasonic sensors that is transferred to the vision system in order to provide an absolute scale for the image coordinates. In [41], the role of the vision system is to track a color target while the ultrasonic sensors are used for collision avoidance. In [42], two cooperative robots are using visual information to detect landmarks for localization. The work in [43] presents robot architecture able to perform real time vision processing using 4 cameras mounted on the robot. Approaches to robot navigation using range measurements derived from ultrasonic sensors are the topic of [44], [28], [45], [46], [47].

The presented approach demonstrates an effective and accurate way how data derived from a vision system can be converted to depth measurements without using any

camera calibration technique. As a case study sonar and vision system data are utilized simultaneously for robot navigation and collision avoidance in indoors environments.

3.3 Vision System

The vision system of the *ATRV-Jr* consists of two uncalibrated pan/tilt color cameras mounted on top of the robot at a distance of *45 cm* from each other. The main steps of the vision algorithm to convert image information to range data are shown in Fig. 3.1(a). The case of monocular vision system differs in the frame grabbing sequence, meaning that the pair of images is grabbed sequentially as illustrated in Fig. 3.1(b).

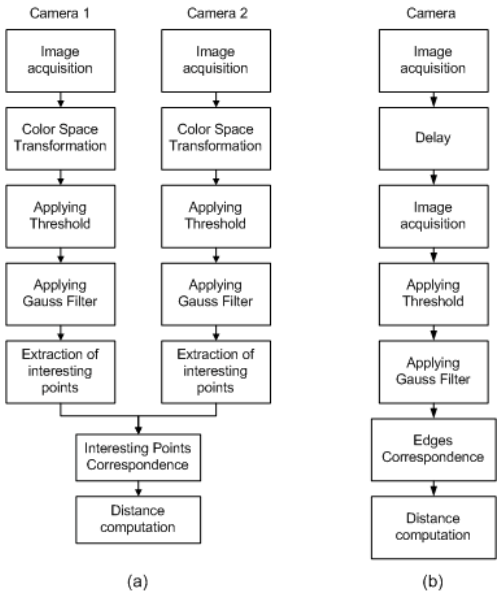


Fig. 3.1 Block diagram of vision system function

3.3.1 Image Acquisition

Image acquisition is achieved using the *Video4Linux API* at a rate of *30 fps*. Since both cameras share the same *frame grabbing* device, the frame rate is reduced to *14 fps* for each camera. Each *24* bit color image has a resolution of *320x240* pixels.

3.3.2 Color Space Transformation

The YC_bC_r color space is used to retrieve the obstacle's color from the acquired image; it is chosen over the HSI and CMYK color spaces because it demonstrated better performance in locating the object in question. The transformation from the original *RGB* color space is documented in [48] and [49].

3.3.3 Applying Threshold

A threshold technique has been implemented in the YC_bC_r, HSI and CMYK color spaces. This is basically done for comparison purposes. Experiments were conducted in an indoors lab environment under different lighting conditions. However the light was uniformly distributed on the obstacles. Images of the same color obstacles obtained in the YC_bC_r, HSI and CMYK color spaces were used for identification and comparison purposes.

At first, an image containing a yellow obstacle was considered, followed by an image containing a multi color obstacle; the purpose of this second image is to illustrate ability to recognize different colors (color segmentation) under different lighting conditions given that the obstacle is located at various distances from the robot.

3.3.4 YC_bC_r Color Space

The C_b component of the YC_bC_r color space is related to the amount of the blue component. Colors containing high quantity of blue color are represented brighter in the C_b image. Yellow contains a narrow quantity of blue color; it is represented as the darkest of all colors in the C_b image. A threshold value to recognize yellow color is:

$$T = t, \text{ when } H(t) > h \text{ and } t = \min(i) \tag{3.1}$$

where T is the threshold value, $H()$ the function of C_b*'s* image histogram, h a number of pixel value that shows when an obstacle is considered big enough to participate in threshold calculation, i is the intensity of the C_b component and t is the minor intensity value corresponding to the number of pixels greater than h. Because of light distribution variations, not every pixel (of the yellow obstacle represented in the C_b image) has the same intensity. A threshold area is used to distinguish the obstacle's pixels from the background, its center being the T value with boundaries between T-$0.1T$ and T+$0.1T$. A new image representing only the object of interest is created according to the equation:

$$g(x,y) = \begin{cases} 255, & T - 0.1T < C_b(x,y) < T + 0.1T \\ 0 & else \end{cases} \tag{3.2}$$

g(x, y) is the intensity function of the new image and $C_b(x, y)$ the intensity function of the C_b component. The value of *T* is computed for each image separately using the image histogram. Thus, the system adapts to illumination variations.

Fig. 3.2 illustrates a color image containing a yellow obstacle (a), the result of the threshold technique (b), and the histogram of a C_b component (c). *T* is the minimum pick on the histogram and the threshold area is *10%* around *T*.

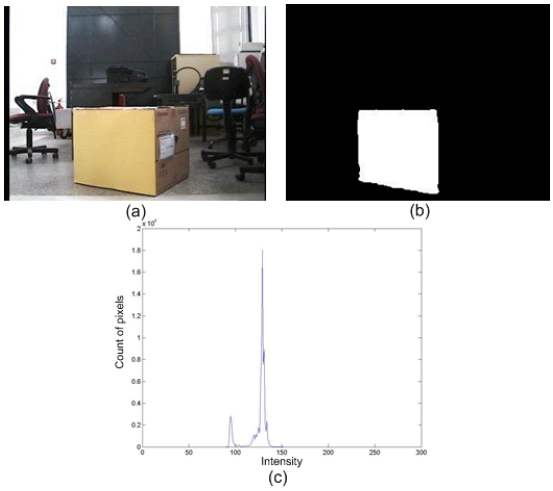


Fig. 3.2 Applying threshold technique in YC_bC_r color space

However, using the properties of the YC_bC_r color space more colors such as red, blue and green may be extracted from a color image. In particular, red color corresponds to the highest intensity values on the C_r component and to medium intensity values between *(130, 160)* on the C_b component. Similarly, blue color corresponds to the highest values on the C_b component and to the lowest values on the C_r component. Finally, green color corresponds to C_b component values in the area between *(70, 100)* and to C_r component values in the area between *(85, 120)*. Fig. 3.3, shows a color image containing

the four color obstacle (a), the result of the threshold technique (b) according to equation (3), and the histograms of the C_b and C_r components.

$$g(x,y)=\begin{cases}255, & C_r(x,y)>T_1 \text{ and } 130\le C_b(x,y)\le 160\\ 200, & 70\le C_b(x,y)\le 100 \text{ and } 85\le C_r(x,y)\le 120\\ 150, & C_b(x,y)>T_2 \text{ and } C_r(x,y)<T_3\\ 100, & T_4-0.1T_4<C_b(x,y)<T_4+0.1T_4\\ 0, & else\end{cases} \tag{3.3}$$

$C_b(x,y)$ and $C_r(x,y)$ are the intensity functions of C_b and C_r components, respectively. T_1, T_2 are the highest picks with the highest intensity value in C_r and C_b components, respectively, T_3 the lowest pick with the lowest intensity value in C_r component and T_4 as defined in (3.1). Implementation using the HSI and CMYK color spaces is described in the appendix.

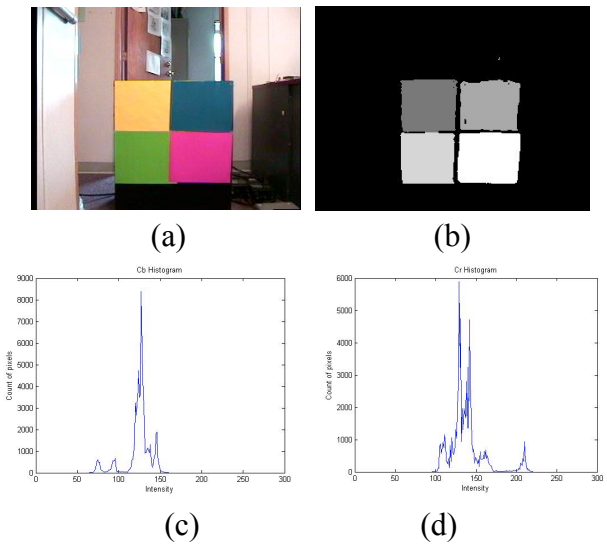


Fig. 3.3 Applying threshold techinque to extract red, green, blue and yellow obstacles using the YC_bC_r color space

3.3.5 Gauss Filter

The thresholded image may have pixels that do not belong to the object of interest. This is because some of the background elements contain a percentage of the color of interest. To eliminate these pixels a *[5x5]* Gauss filter is used [50].

3.3.6 Extraction of Interesting Points

In order to extract characteristic points in an image, the Moravec interest operator has been used [50]. The operator computes the directional variances (I_1, I_2, I_3, I_4) for each point and applies an interest value to it. The directions are computed using all pixels in a window centered about the point, as shown in Equation (3.4). *S* represents pixels of a window centered on this point. The window size used is *[5x5]*.

$$\begin{aligned} I_1 &= \sum_{(x,y)\in S}[f(x,y)-f(x,y+1)]^2 \\ I_2 &= \sum_{(x,y)\in S}[f(x,y)-f(x+1,y)]^2 \\ I_3 &= \sum_{(x,y)\in S}[f(x,y)-f(x+1,y+1)]^2 \\ I_4 &= \sum_{(x,y)\in S}[f(x,y)-f(x+1,y-1)]^2 \end{aligned} \tag{3.4}$$

An interesting value is assigned to each pixel according to:

$$I(x,y) = \min(I_1, I_2, I_3, I_4) \tag{3.5}$$

The image is then segmented in regions; each region's interest point is defined as the pixel with the maximum interest value.

3.3.7 Interesting Points Correspondence

The area of each image that contains the object of interest is segmented in four regions. An interesting point belonging in a region of the first image corresponds to the interesting point belonging in the same region of the second image. The limitation of the method is that it can only retrieve information from the two closest obstacles. However, this method reduces computational time since no correspondence algorithm is needed, which the ATRV-mini's CPU cannot spare. Fig. 3.4 shows the images from left (a) and right (b) camera as well as their corresponding pixels (c).

Whereas the interesting point corespondence mentioned above reports acurate results, it depends on the output of the thresholded image. For this reason, a correlation algorithm may be also used. The correlation coefficient, *r*, is defined as the sum of the products of the pixel brightnesses divided by their geometric mean [51]:

$$r(d_x, d_y) = \frac{\sum_{(i,j)\in S} f(d_x + i, d_y + j).g(i,j)}{\sqrt{\sum_{(i,j)\in S} f^2(d_x + i, d_y + j). \sum_{(i,j)\in S} g^2(i,j)}} \tag{3.6}$$

where *f*, *g* the left and the right image respectively, *S* a pixels window centered around an interesting point (9x9), and d_x, d_y the disparity of the pair of pixels to be matched. In this case all the interesting points of the right thresholded image derived by the Moravec operator are matched with pixels on the left image.

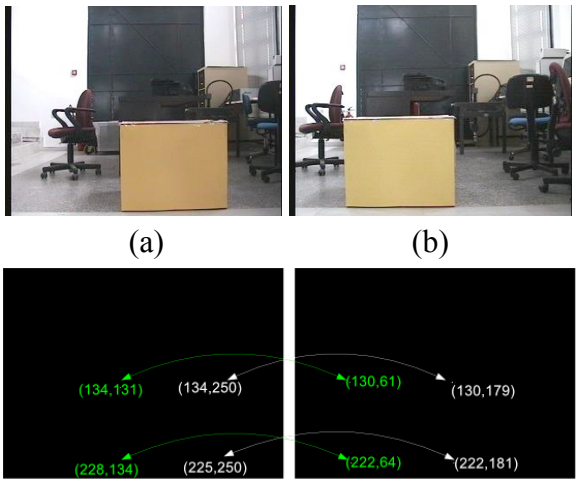


Fig. 3.4 Interesting point correspondence

Both methods have been tested and both provide similar accuracy. For practical reasons, only the first method can be implemented on the ATRV-mini on board computer, in order to get real time results.

3.3.8 Distance Computation

The presented method for depth estimation uses only the size of the image, the angle of view of the camera and the relative position of two different captures of the same scene. Fig. 3.5 shows the field of view of a monocular camera, where A is the length in cm of the horizontal field of view in a distance of Z_i cm from the camera system, (X_i, Yi, Z_i) the world coordinates of any point in the field of view, (x_i, y_i) the projection of the world point onto image plane and (y_p, x_p) its pixel coordinates. The origin of the pixel coordinate system coincides with the center of the image.

For an image of *320x240* pixels, one may observe that:

$$X_i = \frac{2}{320} x_p Z_i \tan(24.4)(cm) \tag{3.7}$$

A second image containing the same world point is acquired from a different position in order to compute depth. In case of a monocular vision system this is achieved by moving towards the world point (stereo from motion). Fig. 3.6 demonstrates this procedure.

For both images the following equations are derived:

$$Z = \frac{x_{p2} B}{x_{p1} - x_{p2}} (cm) \tag{3.8}$$

where B is the relative distance between the two shots in cm (baseline) and x_{p1}, x_{p2} the coordinates of the pixel which represents the world point in the first and second image respectively.

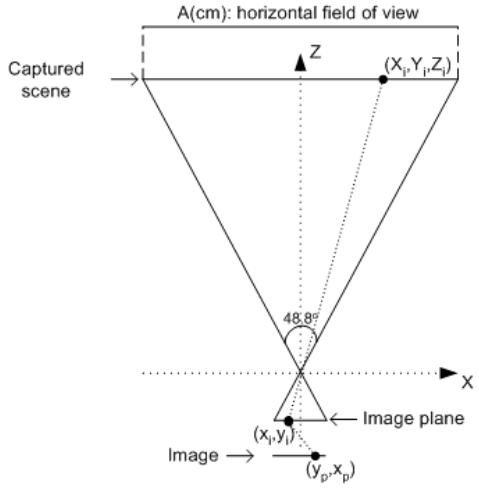


Fig. 3.5 Horizontal field of view of monoscopic vision system

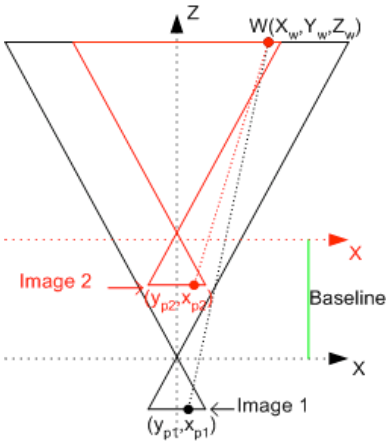


Fig. 3.6 Stereo from motion

For the *parallel* stereoscopic vision system shown in Fig. 3.7 the depth is derived from:

$$Z = \frac{320B}{2(x_{p1} - x_{p2})\tan(24.4)}(cm) \tag{3.9}$$

From Fig. 3.7, it is shown derives that the fields of view of the two cameras converge at a distance of *49.6* cm from the vision system. This is the distance of the closest obstacle that the vision system can locate.

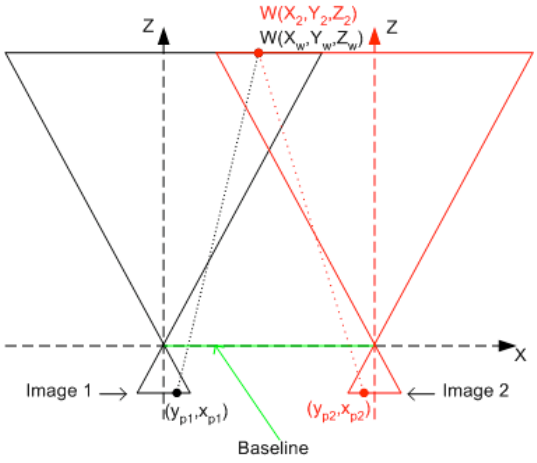


Fig. 3.7 Depth from stereo vision

To minimize this distance, both cameras are rotated such that their optical axes converge closer (see Fig. 3.8). In this work, the angle of rotation, α, is set to 12^o, reducing the distance of the closest obstacle to *27* cm, which is the length between the front bumper of the mobile robot and the base of the vision system. For the rotating stereoscopic vision system of Fig. 3.8, depth is calculated as:

$$Z = \frac{-\cos^2 a - (D-C)\cos a \sin a + CD\sin^2 a}{(D-C)(\cos^2 a - \sin^2 a) - CD\sin 2a - \sin 2a} B \tag{3.10}$$

α is the angle of rotation in degrees,

$$C = \frac{2}{320} x_{p1} \tan(24.4) \tag{3.11}$$

$$D = \frac{2}{320} x_{p2} \tan(24.4) \tag{3.12}$$

Since the area of each image containing the object of interest is segmented into four regions, there are four pairs of corresponding interesting points and, therefore, four measurements of distance. In case of one obstacle the minimum and maximum measurements are ignored and the distance is defined as the average of the other two. In case of two obstacles or one obstacle located at an angle relatively to the vision system, the distance has two values and it is defined as the average of the measurements which correspond to the same vertical edge.

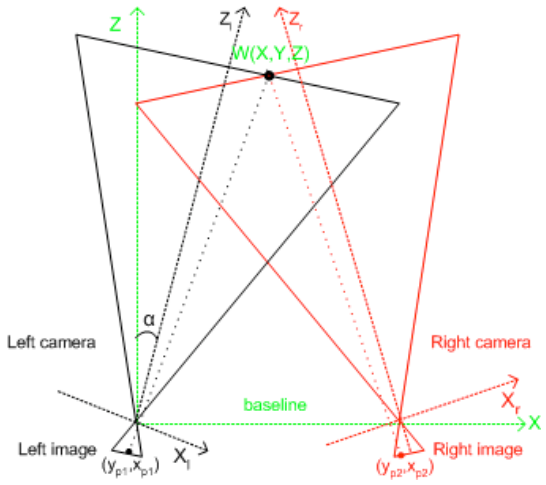


Fig. 3.8 Rotated stereoscopic vision system

3.3.9 Computational Time

The time required converting vision system data to range measurements using the rotating or the parallel vision system is *0.43* sec when running on ATRV-mini's on-board computer, while its speed reaches 0.6m/sec.

3.4 Motion Algorithm

The motion algorithm uses range data from both ultrasonic sensors and the vision system for robot motion planning and collision avoidance. Considering the ATRV – mini, Fig. 3.9, only the eight front ultrasonic sensors have been utilized to derive the motion algorithm. The concept of implemented "collision avoidance" is to calculate the length of the area left and right of the obstacle and to follow the route with larger free space. The

cardinal sensor is the vision system which means that in case both sensors detect an obstacle the range measurements are calculated through vision data.

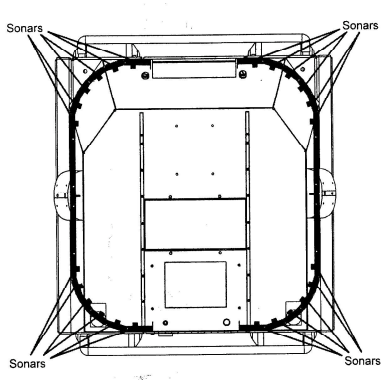


Fig. 3.9 Location of the sonars on board ATRV-mini

Using data from ultrasonic sensors, the free space calculation is done by comparing range measurement summations from sensors mounted on the robot front left and right. If the summation of the left sensor data is greater than the summation of the right one, the robot moves to the obstacle's left, otherwise, it moves to the obstacle's right, according to:

$$\textit{Turn right if: } \sum_{i=0}^{3} U_i < \sum_{i=20}^{23} U_i \tag{3.13}$$

$$\textit{Turn left if: } \sum_{i=0}^{3} U_i > \sum_{i=20}^{23} U_i \tag{3.14}$$

U_i is the range data in *cm* from sensor *i*.

Using data from the vision system, *X,* the horizontal length of an area corresponding to x_p pixels on a *320x240* resolution image is calculated as (see above):

$$X = \frac{2}{320} x \; Z \tan(24.4) \tag{3.15}$$

Z is the distance between the obstacle and the cameras. To calculate the area's length left of the obstacle, x represents the difference in position between the first image pixel and the first obstacle pixel, while for the calculation of the area's length right of the obstacle x represents the difference in position between the last obstacle pixel and the last image pixel such that:

$$X_{left} = \frac{2}{320}(x_p - x_0) Z \tan(24.4) \tag{3.16}$$

$$X_{right} = \frac{2}{320}(x_{320} - x_p) Z \tan(24.4) \tag{3.17}$$

x_0, x_{320} are the first and the last image pixel respectively, and x_p the obstacle's pixel.

Since the vision system is under rotation it is preferred to compute the length of the left area using data from the right camera and the length of the right area using data from the left camera. Collision avoidance is completed by the robot's turn to the side with greater length, until none of the sensors detects an obstacle, and forward movement for Z cm. Fig. 3.10 presents the flow chart of the motion algorithm.

To avoid an obstacle, the robot increases its rotational velocity and reduces its translational speed to half until none of the sensors detects the object. This response guides the robot safely left or right of the obstacle. The current position of the robot is entered to the fuzzy controller [28] and a motion vector reinstates the robot to its previous path towards the goal point.

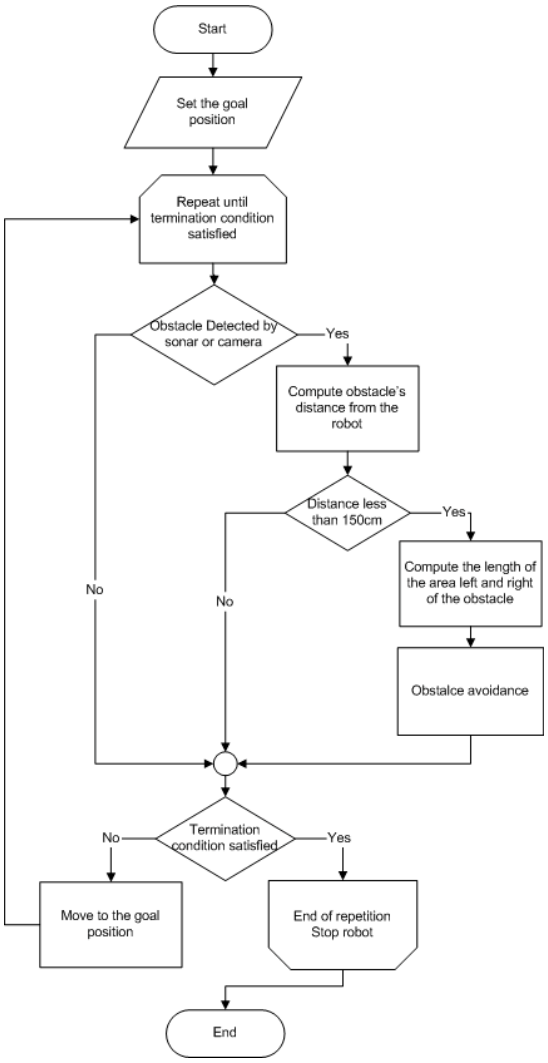


Fig. 3.10 The block diagram of the motion algorithm

3.5 Results

For implementation purposes, two types of obstacles have been used: Type I with size *50x60x30* cm and Type II with size *40x30x20* cm. Type II obstacles are colored yellow and cannot be detected from ultrasonic sensors because of their height, which is equal to the distance between the ground and the ultrasonic sensors. Type I obstacles

cannot be detected from the vision system since they are not yellow. Multicolor obstacles are also used to illustrate capabilities of the approach in identifying different colors at different distances.

Experiments have been conducted in an indoor lab environment with several "furniture obstacles" of different shapes, sizes and orientation, and with different lighting conditions.

3.5.1 Vision System Algorithm

Experimental results showed that the threshold technique in the YC_bC_r color space can identify more than *92%* of the yellow obstacle without detecting any of the background pixels. On the other hand the techniques implemented in the HSI and CMYK color spaces did detect most of the obstacle's pixels (more than *70%* and *85%* respectively), but they also detected many pixels of the background. Hence, the YC_bC_r choice is justified.

Experimental results of range measurements using a parallel and a rotated stereoscopic vision system demonstrate average errors of *2.26%* and standard deviation of *1.1* for the parallel system and *1.58%* and standard deviation of *1* for the rotated one. The relation between the actual and the computed distance is illustrated in Fig. 3.11 and 3.12.

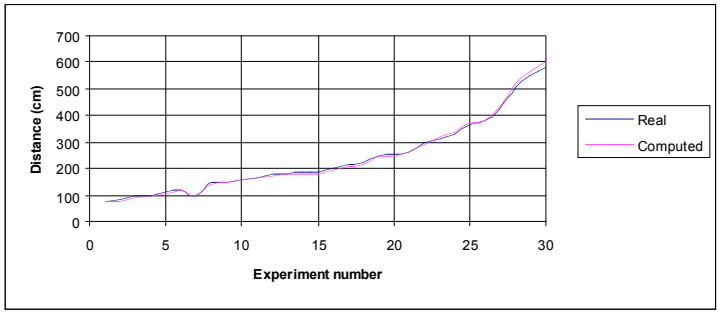


Fig. 3.11 Real distance vs computed distance using data from the parallel vision system

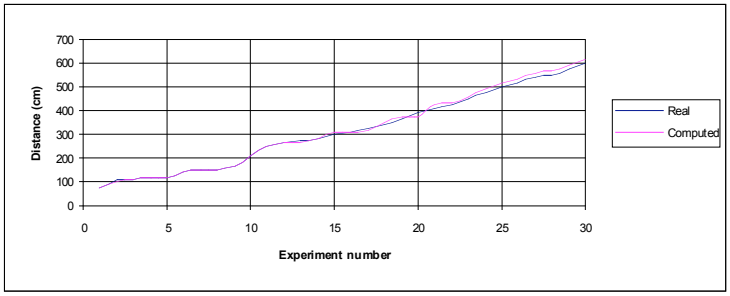


Fig. 3.12 Real distance vs computed distance using data from the rotated vision system

3.5.2 Motion Algorithm

The robot's trajectory is demonstrated from an initial point to a final goal point while collision avoidance is achieved using only the vision system. Fig. 3.13 demonstrates avoidance of two yellow obstacles located in the robot's right, suggesting left movement for avoidance.

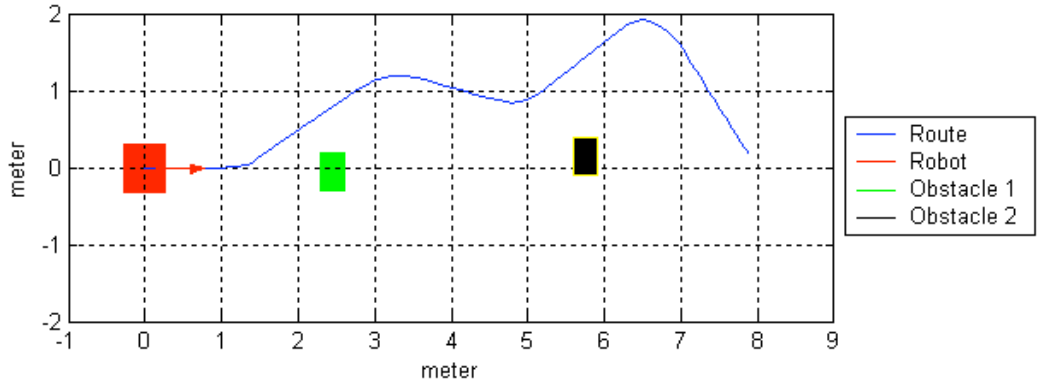


Fig. 3.13 Collision avoidance using vision system data, experiment 1

Fig. 3.14 shows avoidance of two obstacles where the robot turns left because both obstacles are detected from the vision system.

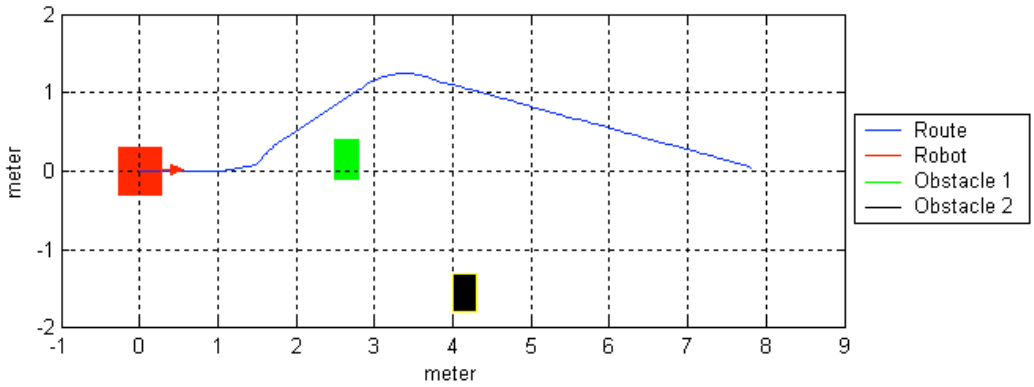


Fig. 3.14 Collision avoidance using vision system data, experiment 2

Fig. 3.15 demonstrates avoidance of three obstacles. The first obstacle that lies in the left of the robot, blocks the vision system from detecting the third obstacle. Thus, the robot has to move left since the vision system can detect the second obstacle.

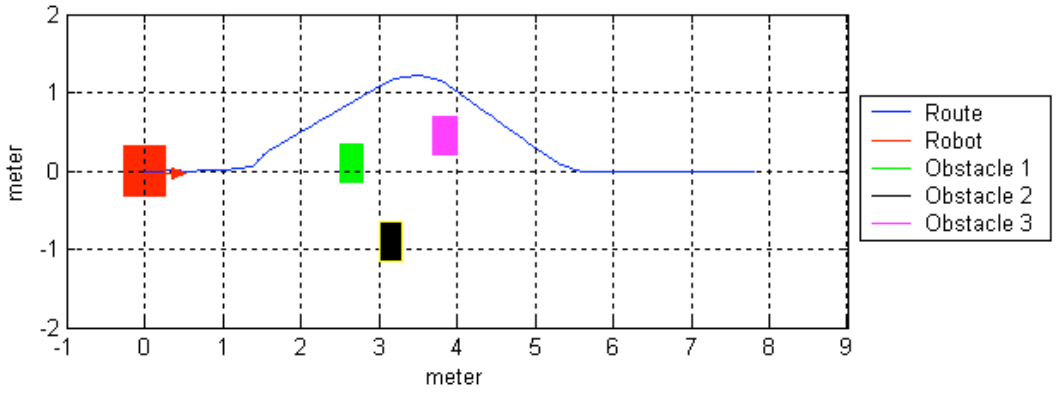


Fig. 3.15 Collision avoidance using vision system data, experiment 3

Fig. 3.16 shows another case with three obstacles. The first obstacle is located right of the robot and it blocks the vision system from detecting the other two. Robot moves left and meets the second obstacle which lies left of the robot. The robot moves right and meets the third obstacle which is located right of the robot. Fig. 3.17 and 3.18 demonstrate the robot trajectory while it avoids three obstacles using range measurements from ultrasonic sensors. To avoid an obstacle, the robot turns left or right according to its relative position with the obstacle.

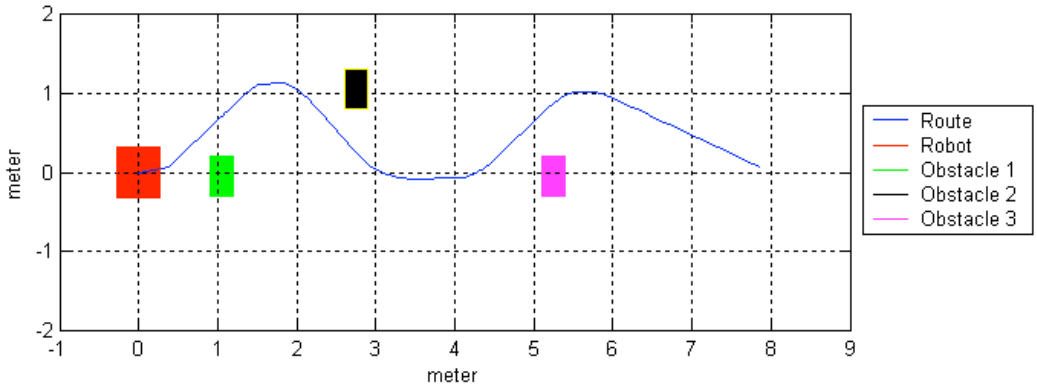


Fig. 3.16 Collision avoidance using vision system data, experiment 4

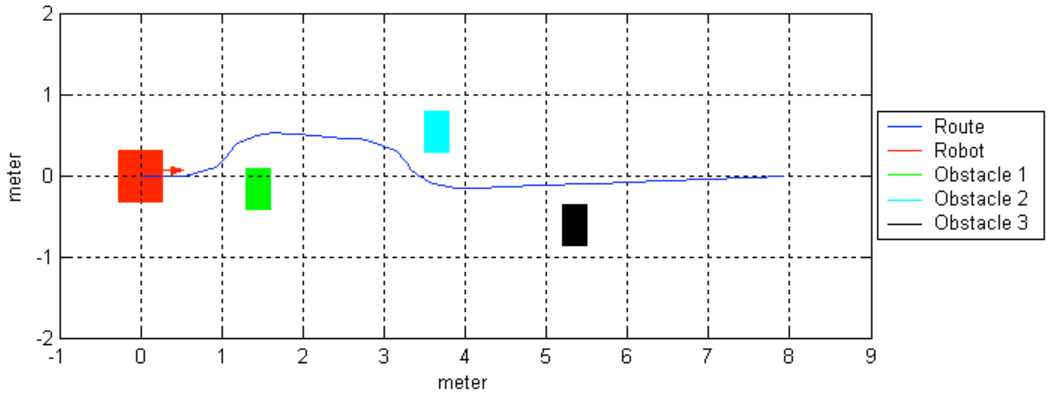


Fig. 3.17 Collision avoindance using range measurments from the ultrasonic sensors, experiment 1

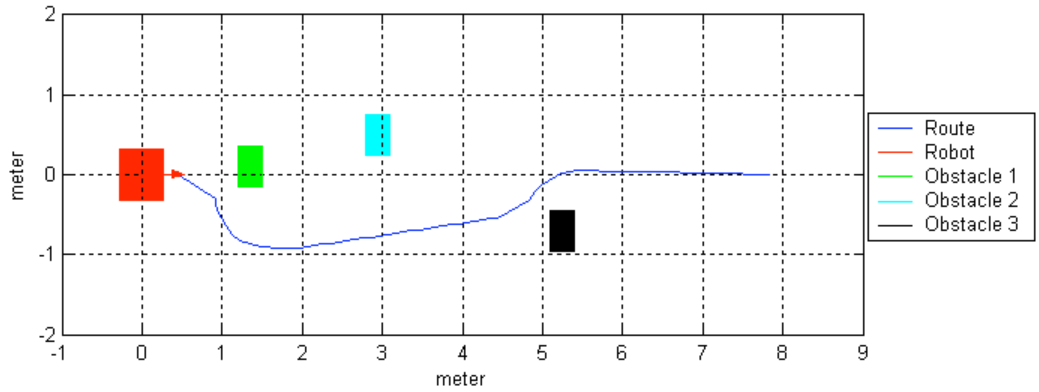


Fig. 3.18 Collision avoindance using range measurments from the ultrasonic sensors, experiment 2

Fig. 3.19-3.22 demonstrate the robot's trajectory and collision avoidance using data from both sonar and camera. For comparison purposes, the robot's trajectory using only the ultrasonic sensors is also shown. Type I obstacles are represented with green color and Type II with cyan.

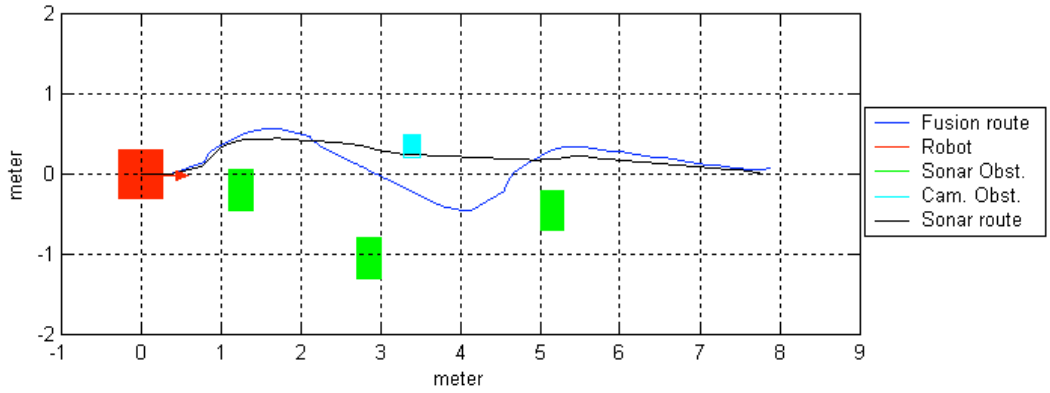


Fig. 3.19 Collision avoidance using data from sonar and camera, experiment 1

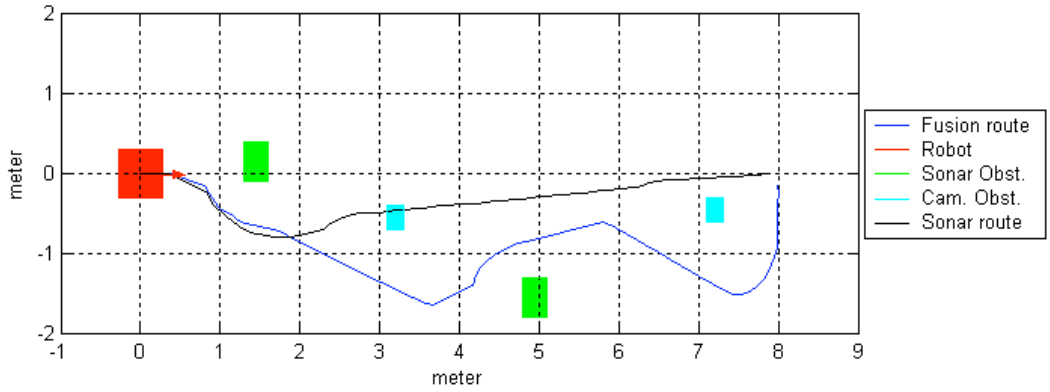


Fig. 3.20 Collision avoidance using data from sonar and camera, experiment 2

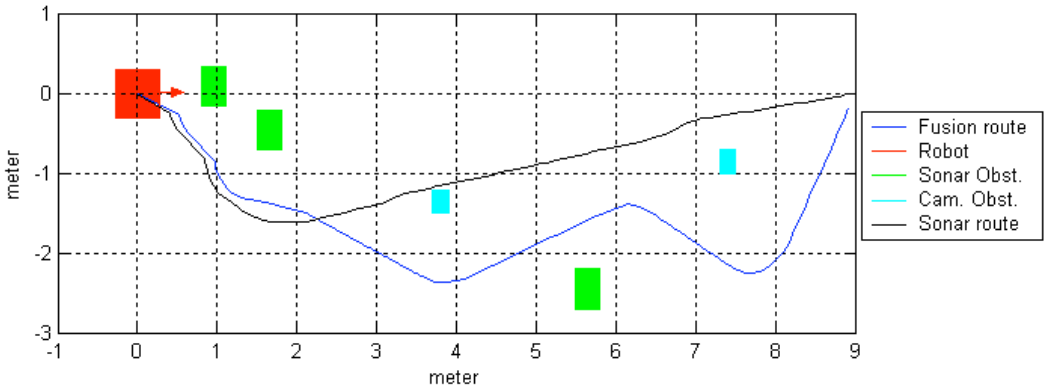


Fig. 3.21 Collision avoidance using data from sonar and camera, experiment 3

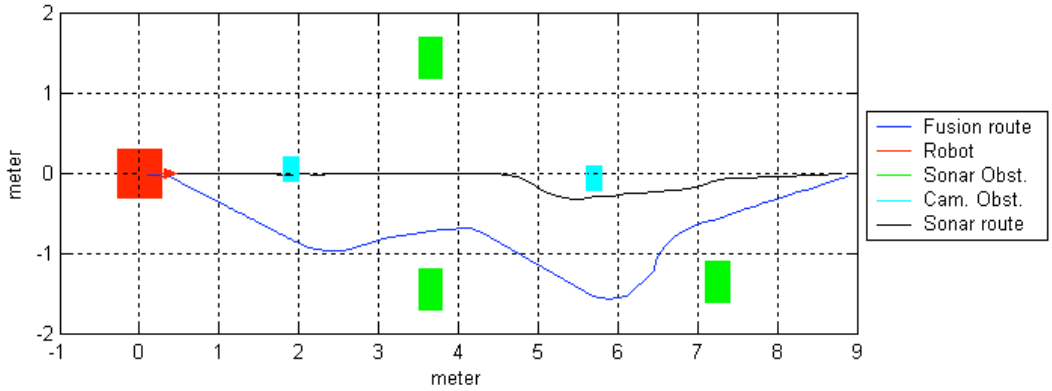


Fig. 3.22 Collision avoidance using data from sonar and camera, experiment 4

Fig. 3.23 shows snapshots of the robot's trajectory while it avoids three yellow obstacles.

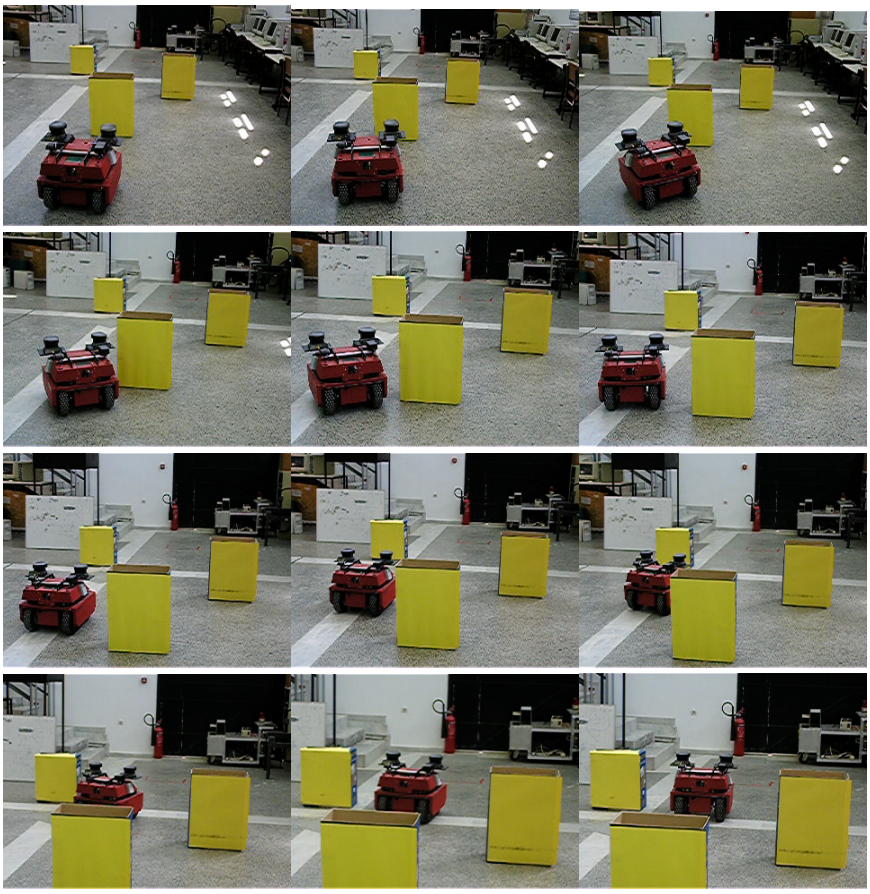


Fig. 3.23 Robot's trajectory snapshots

3.6 Discussion

As mentioned earlier, the case of monocular vision system has also been tested. Fig. 3.1(b) shows the main steps of the vision algorithm where the pair of images needed for range computation is grabbed sequentially. It is assumed that the mobile robot is moving directly towards the object of interest and no rotation is performed during the capture of each image pair. The equation (previously derived)

$$Z = \frac{x_{p2} B}{x_{p1} - x_{p2}} \tag{3.18}$$

is used for distance estimation.

The basic steps of image acquisition, color space transformation, thresholding and filtering remain the same as presented above. However, using the assumption that the mobile robot is moving directly towards the object of interest, the part of interest point correspondence can be improved.

The Moravec Interest Operator increases considerably the computational time of the system (approximately *0.11* sec for each image). Instead, a method of image subtraction can be performed. The pair of the thresholded images is subtracted and the outcome is a new image which reveals the edges of the object captured in the pair of consecutive images (Fig. 3.24). Performing a simple line scan algorithm, the corresponding pixels can be derived.

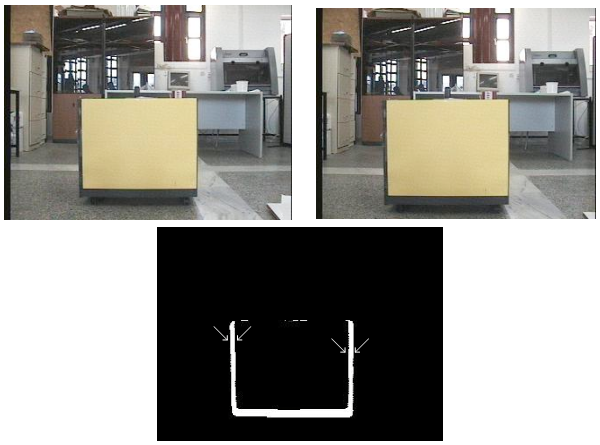


Fig. 3.24 Interesting point correspondence for the case of monocular vision system

The case of the monocular vision system requires accurate knowledge of the distance covered between two consecutive captured images; however, the robot's odometer cannot provide such accurate data. Thus, the computational error is significantly increased. This is the main reason why stereo vision is preferred.

Indicative experimental results on range measurements using a monocular vision system that conducted without the use of the robot's odometer demonstrate mean error of *1.02%* and standard deviation *0.54*.

3.7 Conclusions

This chapter has presented a simple vision based depth estimation technique using the image size of an image, the angle of view of the camera and the relative position of two different captures of the same scene. The effectiveness of the proposed method has

been tested for mobile robot motion planning and collision avoidance in an indoors environment based on simultaneous ultrasonic sensor and/or camera range measurements.

Compared to other related methods, experimental results have confirmed the simplicity and effectiveness of the presented approach as well as superior maximum computational errors and *Normalized Root Mean Square Errors, rmse* of range measurements using either a rotated, parallel, or monocular vision system.

Future work involves sensor fusion between laser scanner and stereoscopic vision system with the introduction of system failures.

Chapter 4

Task Achieving Behaviors

4.1 Introduction

A robust and efficient method is presented for mobile robot dynamic target tracking and collision avoidance in indoor environments using stereo vision and a laser range finder. Key characteristics of the tracking system are real-time operation, efficient computational complexity and ability to adapt to different environments and moving targets. The computational complexity of the vision based tracking and range measurement is $O(n^2)$, where n is the dimension of a square image; the complexity of the laser based range measurement is $O(m)$, where m is the number of laser segments.

Contrary to most existing approaches that are restricted to specific environments and certain types of targets such as templates [52], [53], [54], cars [55], [56] and humans [57], [58], [59], [60], the presented method is rather general and applicable to any indoors environment. The target is identified by its color using the HSI color space and a region growing algorithm. The target color may be either predetermined or dynamically defined, regardless of the target's shape or other physical characteristics. The distance between the target and the mobile robot is calculated using the stereo vision system data.

Limitations have been imposed on the cameras' motion so that pixels correspondences are found across a single epipolar line reducing in this way the computational complexity of the task. Distance is used to control the mobile robot's velocity. Collision avoidance with objects other than the target is accomplished using data from the laser range finder.

The proposed tracking system operates as follows: As soon as a potential target has been identified, both cameras track the target independently of each other with their pan/tilt mechanisms. The distance from the target is calculated using stereo geometry as the mobile robot moves towards the target. The robot's steering angle, φ, is controlled by the angle of the pan mechanism as illustrated in Fig. 4.1. Thus, the target is tracked even if one of the cameras fails to identify it; moreover, in this way, the target is being tracked even while the mobile robot avoids collision with surrounding obstacles, or the target moves in irregular terrains.

Experimental validation and verification of the proposed method is demonstrated using the *ATRV-Jr* skid steering differential drive robot, equipped with *3 GHz P IV* processor and *1 GB* of RAM. The ATRV-Jr uses a *30m* range laser finder that is located in front of the robot, a *GPS*, a compass unit and a stereo vision system that consists of two uncalibrated pan/tilt video cameras, *SONY EVI-D30*, mounted on top of the robot at a distance of *35 cm* from each other. The robot runs *RedHat Linux 7.0* and *Mobility* interface.

Suitable applications that the method may be used are warehouse patrolling and office building security and inspection, where the target may be any moving object and / or human (friend or enemy).

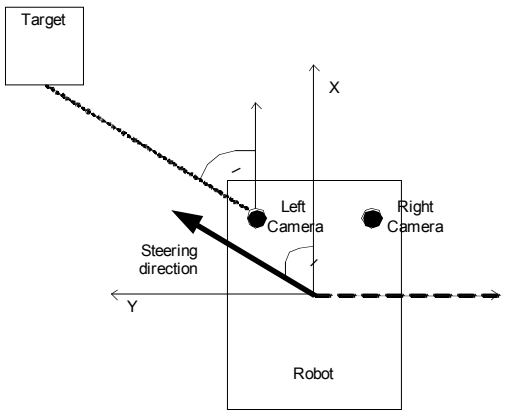


Fig. 4.1 Tracking control system

The chapter is organized as follows: The rest of this section refers to related work and comparisons. Section 4.2 describes the vision system algorithm while Section 4.3 presents the motion algorithm. Section 4.4 is dedicated to experimental results and Section 4.5 concludes the chapter.

4.1.1 Related Research

Reported research in [52] uses template matching techniques for object tracking. The authors use a rated gyro and a camera to solve the problem of image deformation caused by the camera rolling and pitching during the mobile robot's motion on irregular terrains. In [55], a visual feedback controller is proposed using a fixed camera to control the mobile robot to track a moving target. The target is predetermined and its dimensions are known. In [61] and [57] the Condensation Algorithm has been used to implement vision based tracking of moving objects. Objects are tracked by their outlines and features.

Research in [62] solves the problem of target tracking and collision avoidance using magnetic sensors mounted on the left and right of the robot. A multi-modal anchoring method has been used in [58] to track humans with a vision system and a laser range finder. Laser data are used to extract the legs of a person while the skin color is detected through camera images. In [53] a vision based tracking system is presented that uses trinocular sets of precisely aligned and rectified images to represent the trajectory of the object being tracked. In [63] a target tracking controller has been designed with collision avoidance consideration and simulation results are shown. In [56] and [64] an experimental study has been performed on tracking two autonomous mobile robots using only distance sensors. Research in [65] derives a heuristic algorithm to match the velocity of target and tracker in an indoor structured environment. In [59] a person tracking approach is presented using a stereo vision and face detection modules. This system identifies skin color using a converted RGB color space and performs face detection using the face detector library reported in [66]. Reported research in [67] uses infrared sensor to design a fuzzy controller to track a mobile robot. In [68] a fuzzy controller is presented for a general target tracking system. Finally, research in [54] uses template matching techniques for visual target tracking. In [69], a fuzzy algorithm is used to detect an object based on a color cue and tracking is based on a maximum radius of displacement in subsequent frames.

Concerning vision based techniques, research in [70] uses multivariate decision trees for piecewise linear non-parametric function approximation to learn the color of the target object from training samples. The approach in [60] proposes a color model which includes the intensity information in HSI color space using B-spline curves for face track-

ing. Research in [71] uses the Histogram Intersection and Histogram Back-projection techniques to match and locate a color in an image scene. In [72] color segmentation is performed based on contrast information and adaptive thresholds in static images. The authors in [73] use color segmentation to identify obstacles in indoor environments. Using a training set of images and the r-g color space they present a model robust to shadows and illumination changes. Similar to [71] , researchers in [74] propose color indexing using histograms of RBG color ratios. Another color tracking algorithm is presented in [75] where a neural network is used to robustly identify skin color regardless of lighting conditions. In [76] color histogram information is used to detect and track pedestrian for a stationary visual surveillance system.

4.1.2 Comparisons

The main differences and advantages of the presented approach compared to related research are:

1. Using color histograms on the H-I plane allows for a variety of objects to be used as a target, independently of the target's shape, size or other physical characteristics.
2. Both cameras of the vision system track a target independently, providing a redundant mechanism that helps avoiding loosing the target. This means that even if one camera 'loses the target', it can retrieve information from the other camera to find it again.

3. The robot's direction is controlled by the pan/tilt angles of the cameras, allowing the robot to avoid obstacles and keep tracking a target, or to keep tracking a target that moves on uneven terrains.

4.2 Vision System

The vision system of the *ATRV-Jr* consists of two *uncalibrated pan/tilt color cameras* mounted on top of the robot at a distance of *35 cm* from each other. The main steps of the proposed vision based tracking algorithm to convert image information to camera's motion and range data are shown in Fig. 4.2. Each step is presented separately.

4.2.1 Image Acquisition

Image acquisition is achieved using the *Video4Linux API* at a rate of *30 fps*. Both cameras share the same *frame grabbing* device, which supports rates up to *30 fps* from *4* analogue video inputs. Each *24* bit color image has a resolution of *320x240* pixels.

4.2.2 Target Selection

The target is being tracked by its color that can be either predetermined or dynamically defined in the image scene. Identifying color in an image using a region grow-

ing technique instead of template matching or pattern recognition techniques requires less computational time and allows the system to be robust in choosing multiple targets. The *HSI* color space has been chosen for the image manipulation techniques discussed below. This is because the *HIS* color space is similar to the human perception of colors [77].

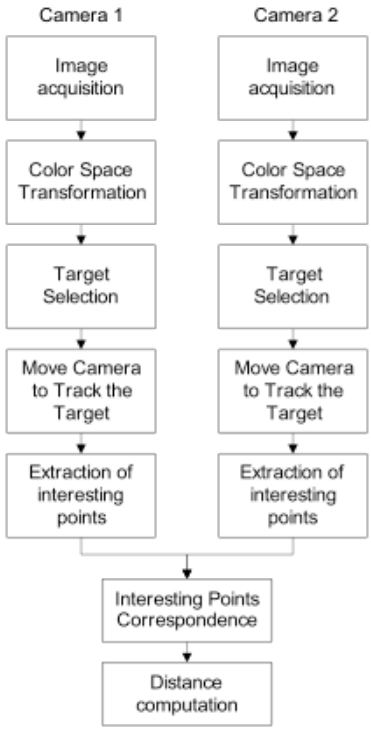


Fig. 4.2 Block diagram of the vision system function

4.2.2.1 Predetermined Color

When the target is a known object, the variation of the *hue*, *saturation* and *intensity* values of the color's representation in the HSI color space is known. A segmentation technique on *H-I* plane based on a region growing algorithm is used to separate the

target from the image's background. The basic concept is to identify a pixel, "seed", in the image that takes the mean *hue* and *intensity* values within the area of the object's color and grow a region with similar neighbouring pixels. For example, the T-shirt in Fig. 4.3a has hue and intensity values that vary between *(310, 340)* and *(125, 145)*, respectively. Thus, the seed pixel will have a *hue* value of *325* and an *intensity* value of *135*. The region growing algorithm will merge neighbouring to the seed pixels that have *hue* and *intensity* values in the former area. Fig. 4.3b shows the result of this technique.

(a) (b)

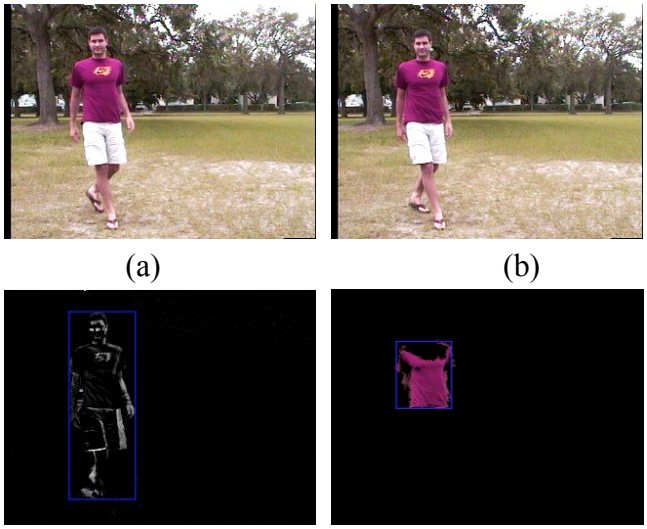
Fig. 4.3 Region growing results for the segmentation of an object with known color

4.2.2.2 Dynamically Determined Color

When the target is unknown, it is defined as the first moving object in the image scene. Motion in a scene is identified by subtracting sequential frames. If the outcome of the subtraction is a black image then no motion has been detected in the image scene. On the other hand, if *2%* or more of the outcome image pixels has values different than zero and it is *4-connected*, then motion has been detected. Initially, the original color images were used for the subtraction, but since the variations of color components are significant

even for images taken within short periods, it has been decided that the subtraction will occur in the greyscale images. The time difference between two frames must be proportional to the target's speed to identify the motion in the images. If the target is moving too fast and the time difference between the two frames is large, then the system will fail to identify motion. Reversely, if the target is moving slow and the time difference between the two frames is small, the threshold of *2%* of the image pixels will not be met. Experimentally it has been determined that for a walking man or for a moving mobile robot a difference of *0.3sec* is adequate to identify motion between the two frames. A median filter with window size *5x5* is applied to the subtracted image to eliminate individual pixels that were erroneously recognized as 'motion'. These pixels usually belong to objects with shining surfaces where light is irregularly reflected. When the target has been detected, the RGB to HSI transform is performed to the region of the image that surrounds it. This region will be denoted from now on as *region of interest*.

To identify the color of the moving object, the histograms of the *hue* and *intensity* components are computed for the region of interest. The *hue* and *intensity* value with the greater frequency in the region of interest is used as the seed value for the region growing algorithm. To improve the accuracy of this selection, the seed is the pixel where its *8*-connected neighbours present the maximum *hue* and *intensity* values in the region of interest. This is a gruelling criterion for the selection of the seed that helps to avoid locating objects in the scene with similar colors. Then, the region growing algorithm is applied to merge pixels that present *hue* and *intensity* values in the range of ± 20 of the seed's values. In Fig. 4.4 two sequential images (a), (b) are depicted as well as the region of interest derived by the motion (c) and the region growing algorithm (d).



(a) (b)

(c) (d)

Fig. 4.4 Region growing results for the segmentation of an object with unknown color

The *hue* and *intensity* component histograms are only computed for the initial pair of images and the maximum values are used to run the region growing algorithm for the rest of the images. Since illumination may vary even in indoor environments the maximum values of the Hue and Intensity components are recalculated for the region of interest in the case that the seed cannot be found.

4.2.3 Target Tracking

One of the differences of the proposed method compared to others is that in existing approaches there is an effort to control the tracker's speed and steering angle to follow the target holding the vision system fixed. In the presented method, each camera of the stereo vision system tracks the target using their pan/tilt mechanisms. Thus, the target is being tracked even when the robot is in collision avoidance mode, or when the target is

moving in irregular terrains. The velocity of the robot is adjusted according to the relative distance between the robot and the target, calculated using data derived from the stereo vision system. The robot's steering angle, φ, is controlled by the angle of the pan mechanism of the cameras, Fig. 4.1.

Each camera's motion is such that it keeps the target at the center of the image. Therefore, each image is segmented into *14* regions as shown in Fig. 4.5. When the mass center of the pixels belonging to the target falls into a numbered sub image, an appropriate motion of the pan/tilt mechanism tends to reinstate the target to the center of the image. Experimentally it has been determined that when the mass center resides to the sub images 1-8, an angle of approximately 7^o on the horizontal axis and 5.4^o on the vertical axis is efficient to reinstate the target to the center of the image. When the mass center resides on the sub images 9-*14* a sharper motion is required, thus the angle of the horizontal axis is set to 12^o.

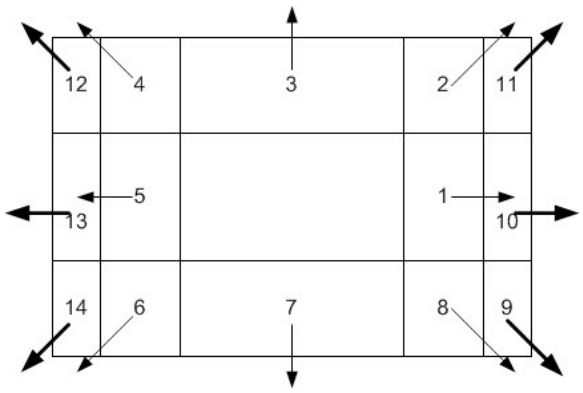


Fig. 4.5 Image segmentation for camera motion

Since stereo correspondences are needed to calculate the distance between the robot and the target, both cameras are forced to the same angle in the vertical axis (tilt). This allows calculating pixel correspondences across a single *epipolar line* instead of the whole images, reducing the required computational time.

The case of one camera failing to locate the target is also considered; then, data derived and collected from the failed camera may no longer be useful. Two cases have been considered. First, the target is located in front of the robot Fig. 4.6a, and second the target is located on an angle with respect to the robot's direction, Fig. 4.6b. In the first case the failed camera will be forced to the opposite angle of the second camera and in the second case the failed camera will be forced to the angle of the second camera.

Given the robot's velocity and the camera's angle on the horizontal axis, the steering velocity of the robot can be computed:

$$| \dot{y} | = | \dot{x} | \tan\phi \tag{4.1}$$

where y is the steering velocity, x is the robot's velocity and φ is the horizontal angle of the camera. The robot's velocity depends on the distance between the robot and the target and it varies from *0-1* m/s.

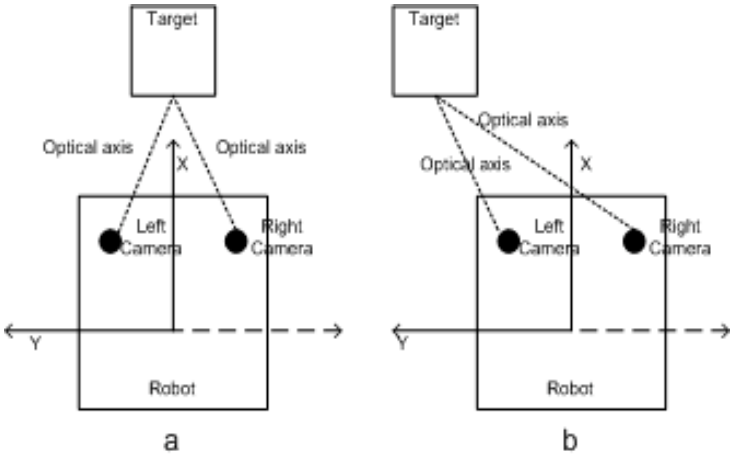


Fig. 4.6 Location of the target related with the robot

4.2.4 Extraction of Interesting Points

In order to extract characteristic points in an image, the Moravec interest operator has been used [50]. The operator computes the directional variances (I_1, I_2, I_3, I_4) for each point and applies an interest value to it. The directions are computed using all pixels in a window centered on the point, as shown in (2). *S* represents pixels of a window centered on this point. A window size of *5x5* is used.

$$
\begin{aligned}
I_1 &= \sum_{(x,y)\in S} [f(x,y) - f(x,y+1)]^2 \\
I_2 &= \sum_{(x,y)\in S} [f(x,y) - f(x+1,y)]^2 \\
I_3 &= \sum_{(x,y)\in S} [f(x,y) - f(x+1,y+1)]^2 \\
I_4 &= \sum_{(x,y)\in S} [f(x,y) - f(x+1,y-1)]^2
\end{aligned}
\tag{4.2}
$$

An interest value is assigned to each pixel according to:

$$I(x,y) = \min(I_1, I_2, I_3, I_4) \tag{4.3}$$

The image is then segmented in regions where each region's interest point is defined as the pixel with the maximum interest value.

4.2.5 Interesting Points Correspondence

A correlation algorithm is used to solve the *correspondence problem*. The correlation coefficient, *r*, is defined as the sum of the products of the pixel brightness divided by their geometric mean [51]:

$$r(d_x, d_y) = \frac{\sum_{(i,j)\in S} f(d_x + i, d_y + j).g(i,j)}{\sqrt{\sum_{(i,j)\in S} f^2(d_x + i, d_y + j). \sum_{(i,j)\in S} g^2(i,j)}} \tag{4.4}$$

where *f*, *g* are the left and the right image respectively, *S* is a 9x9 pixels window centered on an interesting point, and d_x, d_y are the disparity of the pair of pixels to be matched. Since both cameras are forced on the same vertical angle, the assumption of single epipolar line is valid. This means that every interesting point generated by the Moravec operator from the right image that belongs to a certain row has to be correlated with interesting points of the left image in the same row. Experimentally it has been determined that the pairs of pixels that have correlation coefficient greater than 0.99 are considered corresponded.

4.2.6 Distance Computation

The method that is used to compute the distance between the robot and the target requires the angle of horizontal view of the camera as the only parameter for depth estimation. Fig. 4.7 shows the field of view of a monocular camera, where *A* is the length in *cm* of the horizontal field of view in a distance of Z_i *cm* from the camera system, (X_i, Yi, Z_i) is the world coordinates of any point in the field of view, (x_i,y_i) is the projection of the world point onto image plane and (y_p, x_p) are the pixel coordinates. The origin of the pixel coordinate system coincides with the center of the image.

For an image of *320x240* pixels, one may observe that:

$$X_i = \frac{2}{320} x_p Z_i \tan(24.4)(cm) \tag{4.5}$$

A second image containing the same world point is acquired from a different position in order to compute depth. In case of a monocular vision system this is achieved by moving towards the world point (stereo from motion). Fig. 4.8 demonstrates this procedure. For both images the following equations are derived:

$$Z = \frac{x_{p2} B}{x_{p1} - x_{p2}} (cm) \tag{4.6}$$

where B is the relative distance between the two shots in cm (baseline) and x_{p1}, x_{p2} are the coordinates of the pixel which represents the world point in the first and second image respectively.

For the *parallel* stereoscopic vision system shown in Fig. 4.9 the depth is derived from:

$$Z = \frac{320B}{2(x_{p1} - x_{p2})\tan(24.4)}(cm) \tag{4.7}$$

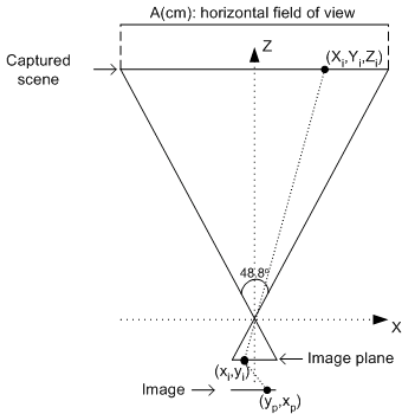


Fig. 4.7 Horizontal field of view of monoscopic vision system

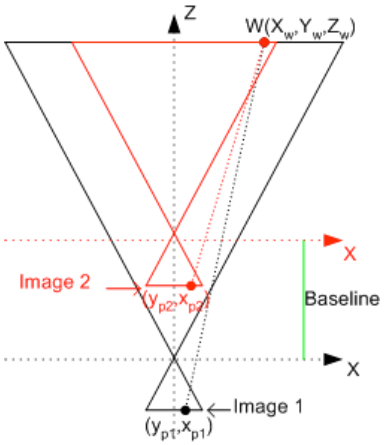


Fig. 4.8 Stereo from motion

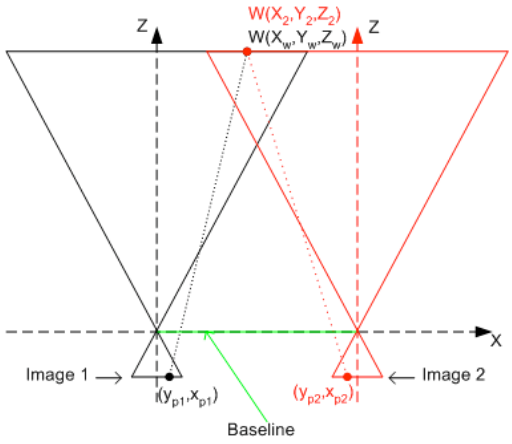


Fig. 4.9 Depth from stereo vision

Since both cameras are moving independently there are 3 angles that have to be considered for the depth estimation; one for each pan and one for the tilt of the cameras. For this rotating stereoscopic vision system depicted on Fig. 4.10, depth is calculated as:

$$Z = \frac{320\cos(c)}{2\tan(24.4)[\cos(a)x_{p1} - \cos(b)x_{p2}] + 320[\sin(b) - \sin(a)]}B(cm) \tag{4.8}$$

where α and b are the horizontal angles of rotation in degrees of the left and the right camera respectively, c is the common vertical angle of rotation and B is the base line.

4.2.7 Computational Time

To reduce the computational complexity of the algorithm, distance calculations are not performed for each pair of frames. This action is executed every *30* pairs of frames and only if both cameras have identify the target.

The computational time required to convert visual information to range measurements depends heavily on the part of the image that the target covers. The maximum frame rate reaches the *20* fps. This is an adequate frame rate when the target is a walking man or a mobile robot, considering that the maximum velocity of the tracker cannot exceed the *1* m/s.

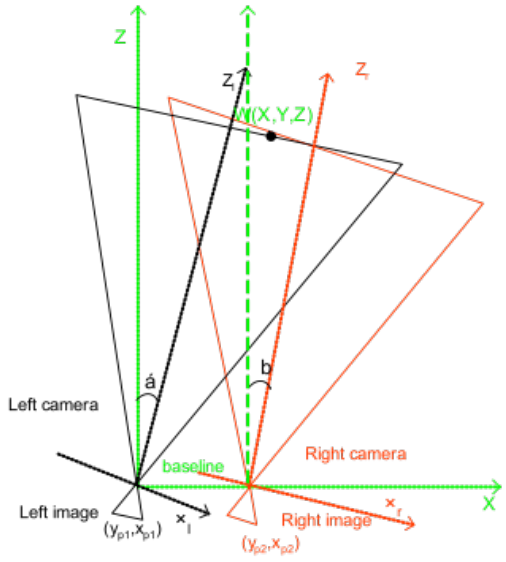


Fig. 4.10 Rotated stereoscopic vision system

4.3 Motion Algorithm

The laser range finder is divided into three regions; each one is responsible for collision detection in the three main directions: *front*, *right* and *left*. A fourth range measurement is computed by the stereo vision system and it is associated with the distance of the target from the robot. All these inputs are fed to the motion algorithm, which decides for the vehicle's rotational and translational speed. Fig. 4.12 depicts the flow chart related to the motion algorithm.

The assumption is that the robot stays still until the vision system detects a target. This is a valid assumption since a failed robot will remain in its current location until it receives new commands from a supervisory controller, or a patrolling robot will visually inspect its territory until a motion is detected.

As soon as the target has been identified and its distance from the robot has been computed, the robot starts moving towards the target. Assuming that the robot has to reach the maximum velocity when its distance from the target is more than *4m*, then the robot's velocity is linearly derived by:

$$|\bar{x}| = Z/4 \tag{4.9}$$

where $\bar{x}$ is the robot's velocity and Z is the distance derived from the vision system.

Given the robot's velocity and the camera's angle on the horizontal axis, the steering velocity of the robot is obtained using (4.1).

Collision possibilities are computed using the data derived from the laser range finder in the three main directions, front, right and left of the robot, as shown in Fig. 4.11.

The safety distance from the robot has been arbitrarily set to *40 cm*. When an obstacle is detected at a distance less than this threshold; a correction in the steering angle of the robot is performed driving it away from the obstacle. At the same time both cameras keep tracking the target regardless of the robot's motion. When the vehicle passes the obstacle, it resumes its past route according to the cameras' pan positions. The system stops the robot when the distance computed by the vision system is less than *1m*.

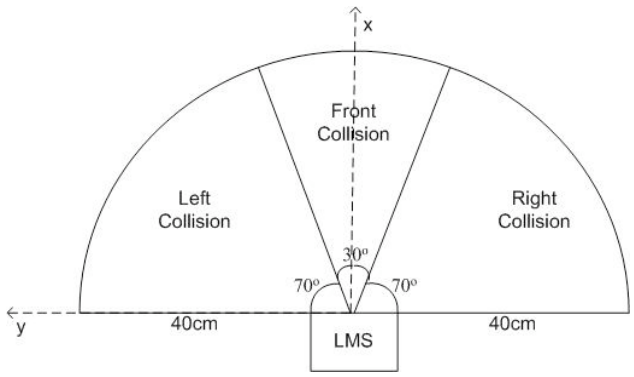


Fig. 4.11 Segmentation of the laser finder beam in three regions

The correction in the steering angle forces the robot to keep a distance greater than *40 cm* from the obstacle, while the velocity of the robot reduces to half. This is performed using a simple set of rules:

1. If an obstacle is detected in the right side of the robot, turn left until its distance is greater than 40cm. Then carry on the target's route.
2. If an obstacle is detected in the left side of the robot, turn right until its distance is greater than 40cm. Then carry on the target's route.
3. If an obstacle is detected both in the left and the right side of the robot and the projections on the *y* axis (Fig. 4.10) of both distances are greater than 20cm, which is the distance between the center of the laser range finder and the edges of

the robot, then continue straight passing through the obstacles. Otherwise, conventionally, turn right until the distance computed by the left segment of the laser is greater than 40cm.

4. If an obstacle is detected in front of the robot, it is unlikely that this object is the target since the robot would have stopped moving, turn right until the distance computed by the left segment of the laser is greater than 40cm.

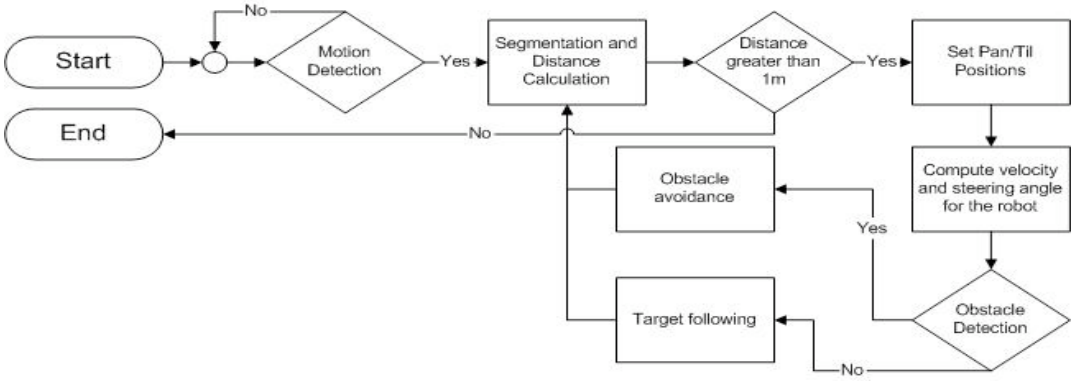


Fig. 4.12 Flow chart of the motion algorithm

4.4 Results

Experiments have been conducted in an indoor lab environment with several "furniture obstacles" of different shapes, sizes and colors. For implementation purposes a person wearing different color clothes is used as the target. In the frames that follow, one can note the difference in lighting conditions, as well as the ability of the algorithm to

distinguish the target from the background, even if objects with similar colors appear in the image scene.

Fig. 4.13 demonstrates the ability of the vision system to detect the target wearing blue pants. The robot in this sequence is not moving. Fig. 4.14 demonstrates the same procedure but this time a red color is targeted. Notice that the red cones on the background does not confuse the tracking system.

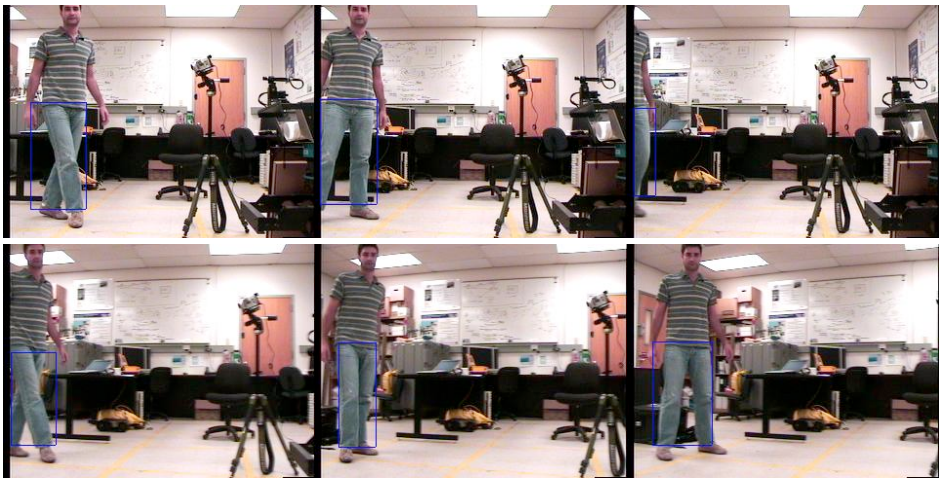


Fig. 4.13 Tracking blue color

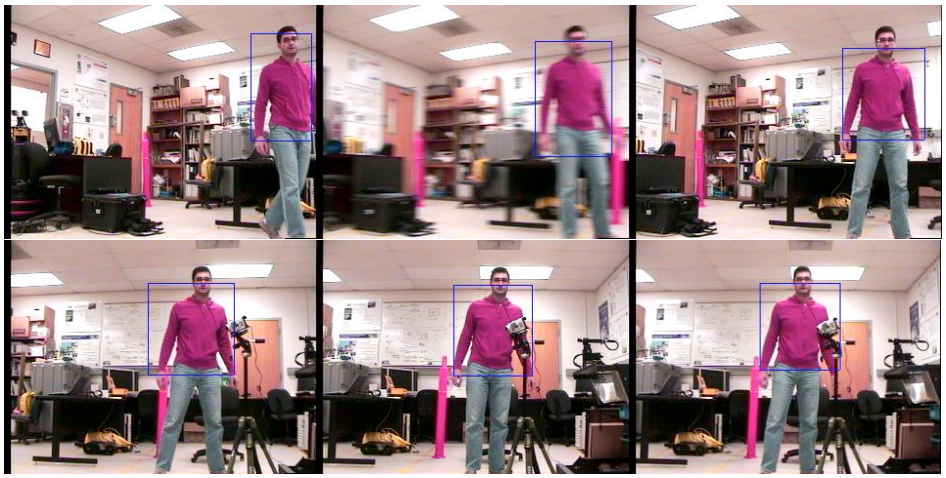


Fig. 4.14 Tracking red color, example 1

Fig. 4.15 presents the robot's view as it follows a person in a corridor. Notice the cameras' motion and the turn of the robot at the end of the corridor. Fig. 4.16 and 4.17 demonstrate tracking and obstacle avoidance from an external video source.

Fig. 4.15 Tracking red color, example 2

Fig. 4.16 Tracking red color and obstacle avoidance

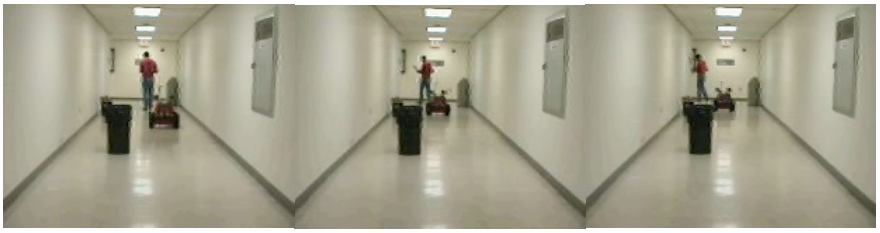


Fig. 4.16 (Continued)

Fig. 4.17 Tracking blue color and obstacle avoidance

Fig. 4.18 demonstrates the ability of the vision system to detect a mobile robot that passes in front of the tracker.

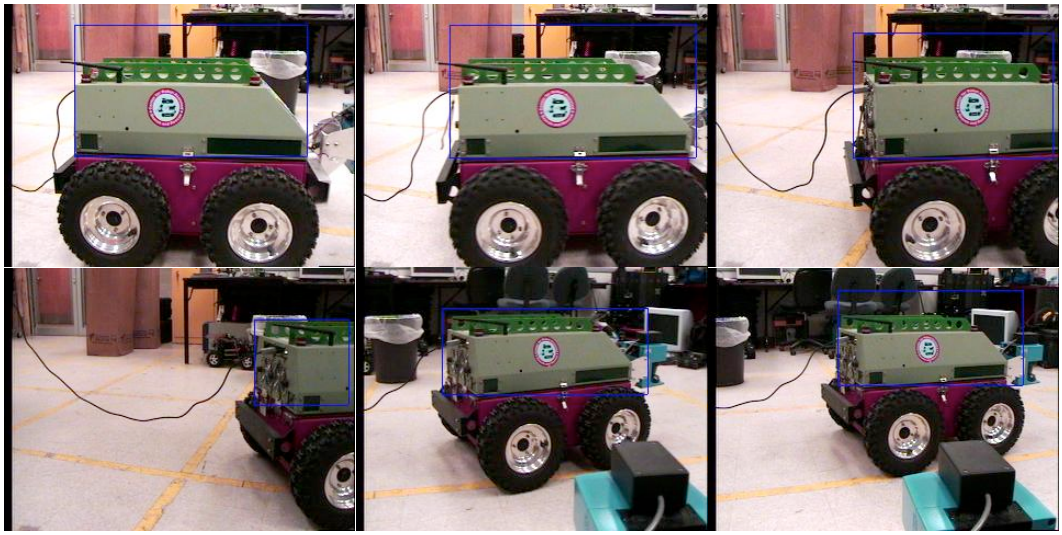


Fig. 4.18 Tracking a mobile robot

Fig. 4.18 (Continued)

4.5 Conclusions

This chapter has presented a vision based target tracking method for mobile robots using a stereoscopic vision system and a laser range finder. Experimental results demonstrate the effectiveness and the robustness of the approach.

Significant advantages over the other vision based target tracking techniques concern the ability on tracking a variety of objects and that the robot's motion derives from the horizontal angle of the cameras, which allows the robot to avoid obstacles and keep tracking a target, or to keep tracking a target that moves on irregular terrains.

The presented approach is initialized using motion estimation techniques to specify the target and compute the color histograms. In the presented results the target is the only object moving in the image scene. However, if more than one object is moving, the algorithm can be trained for the one that occupy greater portion of the image.

Alternative methods to avoid the randomness of the motion estimation can be used. Methods such as template matching will specify with accuracy the object that needs to be tracked. This is only for initialization purposes. When the object of interest is located in the image, its color histograms are computed and the rest of the algorithm continues as is.

Future work involves vision based target tracking in outdoor environments where the variance in the color components is significant.

Chapter 5

Localization

5.1 Introduction

Localization is prerequisite to Unmanned Ground Vehicles (UGV) accurate and autonomous navigation in uncertain and/or unknown environments. At any given time, the UGV should have exact knowledge, or should be able to derive, or estimate accurately its position and orientation.

Required information is basically acquired from position estimation sensors such as Global Positioning System (GPS), Inertial Measurement Unit (IMU) and odometers. The individual use of these sensors has specific disadvantages. For example: GPS cannot provide accurate position estimation when the UGV travels small distances or it travels in indoor environments; IMU is sensitive to sadden accelerations and magnetic interferences; odometers lose track of the UGV due to wheel slippage. Thus, a method that fuses data and information acquired from multiple sensors providing an accurate estimation of the UGV position and orientation is required.

This chapter proposes a multi sensor fusion method for UGV localization based on Fuzzy Logic (FL) and Extended Kalman Filters (EKF). The sensors considered are a GPS,

an IMU, a stereo vision system, a laser range finder, and the vehicle's odometer. All sensors are mounted on top of the UGV. Position sensors such as GPS and odometer provide information related to the vehicle's posture. Inertial sensors such as the IMU compute accelerations and angular rates. Finally, range sensors such as the stereo vision system and the laser range finder compute distances between landmarks and the vehicle. The FL controllers proposed in this chapter receive inputs such as sensor readings and environmental characteristics to compute the statistics of the error distributions in the measurements of the GPS, the odometer, the vision system and the laser range finder. In this chapter, the standard definition of error is used: the difference between the true value and the measured value divided by the true value.

In contrast to the traditional EKF approach where the error in sensor readings is either preset or an additional variable in the system model, the proposed fuzzy EKF approximates the error in sensor readings with multiple zero mean Gaussian distributions, based on the performance of the sensor in a series of experiments. In this way, the error in sensor readings is incorporated into the covariance matrix $\mathbf{R}$ of the measurement's model as zero mean Gaussian distribution and not as an unknown variable in the system's model.

Experimental validation and verification of the proposed method is carried out using the *ATRV-Jr* skid steering differential drive UGV, equipped with 3 GHz P IV processor and 1 GB of RAM. The sensor suit of the *ATRV-Jr* consists of a SICK laser range finder, a GPS, a Microstrain IMU, a compass unit and a stereo vision system. The vehicle runs on RedHat Linux 7.0 and Mobility interface. The experiments have been conducted indoors and outdoors, on different types of floors and different weather conditions.

The rest of the chapter is organized as follows: Section 5.2 discusses work related to the UGV localization problem and summarizes the contributions of proposed research. Section 5.3 presents the EKF while Section 5.4 discusses the FL extension of the EKF. Section 5.5 demonstrates the use of the fuzzy EKF, while Section 5.6 concludes this chapter.

5.2 Related Work

The EKF has been widely used in mobile robot navigation applications, mostly to integrate data from position sensors such as GPS, INS and odometer, and/or range sensors such as laser range finder and ultrasonic sensors. The choice of the sensors to be integrated is related to the robot's usage and operational environment. For example, indoor applications may not benefit from a GPS unit due to signal scrambling and outdoor applications may not benefit from ultrasonic sensors due to their limited performance outside.

Examples of sensor integration for mobile robot localization include odometry and laser [78], [79], [80], [81]; odometry and ultrasonic sensors [82]; odometer, gyroscope and GPS [83]; INS [84]; INS and vision system [85]; GPS, INS, odometer and vision system [86]; vision system and odometry [87], [88], [89], [90], [91]; laser and vision system [92], [93], to name a few.

A drawback of the EKF is that it requires precise knowledge of the system's dynamics, and noise distributions to be Gaussians with zero mean [94]. Various techniques have been proposed to overcome these problems with some focusing on the use of FL. Re-

search in [95] proposes an adaptive fuzzy Kalman Filter (KF) method for mobile robot localization. A weighted EKF is used to fuse measurements from a GPS and an INS, while a fuzzy logic system is used to tune weights based on the covariance and the mean value of the residuals. In this way, the EKF is prevented from diverging. In [96], two FL controllers are used to enhance the EKF. The first adjusts the measurement covariance matrix based on a covariance matching technique and the second monitors the performance of the EKF and assigns a degree of confidence on the outputs of the first fuzzy controller. In [97], a Takagi-Sugeno fuzzy model is used to linearize the EKF system model. In [98], a fuzzy KF is proposed based on the assumption that the measurement and the model uncertainty is 'possibilistic' instead of Gaussian. In [99], a Neuro-Fuzzy KF is presented to evaluate sensor measurements and measurement errors. Finally, in [95] an adaptive fuzzy logic system is discussed that modifies the Kalman gain preventing the filter from diverging.

The contribution of this research can be summarized as follows:

1. The localization method presented fuses information derived from five different sensors: GPS, IMU, odometer, stereo vision system and laser range finder, while most related work utilizes up to three sensors.
2. A framework that handles data and information from multiple range sensors and multiple landmarks is provided.
3. The FL controller design allows for real time computation of the statistics of the error in sensor readings used in the EKF without prior knowledge of the mathematical model of the sensor.

4. An important characteristic of the presented method is that the landmarks' positions with respect to the vehicle or any other general coordinate system is not required.

5.3 Extended Kalman Filter

The EKF is a powerful estimation tool designed for systems described by non-linear equations. To implement the EKF it is necessary to define models describing the system's dynamics and sensor measurements. The rest of this section describes the development of the EKF models.

5.3.1 Constructing the System's Model

In localization applications, the system's model represents the belief of the vehicle's posture based on a series of mathematical equations such as the kinematics equations, landmark positions or error models. The kinematics equations for a differential drive vehicle, which provide an estimate of the vehicle's posture, are defined as:

$$\mathbf{x}_{k+1} = \begin{bmatrix} x_k + V_k \Delta t \cos \vartheta_k + \frac{1}{2} a x_k \Delta t^2 \\ y_k + V_k \Delta t \sin \vartheta_k + \frac{1}{2} a y_k \Delta t^2 \\ \vartheta_k + \Omega_k \Delta t \end{bmatrix} \tag{5.1}$$

where $\mathbf{x}_k = \left[x_k \; y_k \; \vartheta_k\right]^T$ is the vehicle position at time k with respect to a global coordinate system, $\mathbf{u}_k = \left[V_k \; \Omega_k\right]^T$ is the control input consisting of the vehicle linear and angular velocities and ax_k, ay_k the vehicle's accelerations on axes x and y respectively (Fig. 5.1). It is assumed that the vehicle travels in a 2D environment. It is also assumed that the accelerations ax_k, ay_k are constant:

$$\begin{aligned} ax_{k+1} &= ax_k \\ ay_{k+1} &= ay_k \end{aligned} \tag{5.2}$$

In addition to vehicle's kinematics, range readings derived by sensors as the laser range finder, provide additional means of estimating the vehicle's position. Consider a set of landmarks in positions, $\mathbf{lan}_k^i = \left[lan_{x,k}^i \; lan_{y,k}^i\right]^T, i = 1...n$, with respect to a global coordinate system as depicted in Fig. 5.1. These landmarks are stationary and their position in time can be expressed as:

$$\mathbf{lan}_{k+1}^i = \mathbf{lan}_k^i \tag{5.3}$$

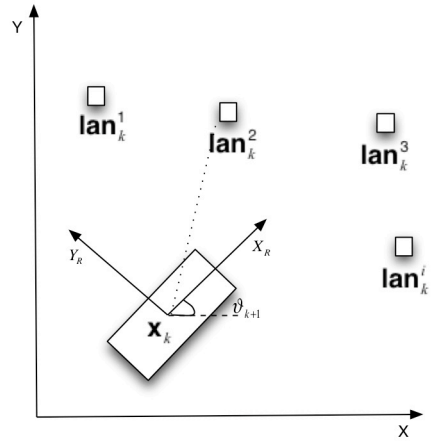


Fig. 5.1 Landmark position with respect to a global coordinate system

Equations 5.1, 5.2 and 5.3 are used to describe the system's model for most UGV localization applications. The system's model is described by the equation:

$$\mathbf{X}_{k+1} = \mathbf{f}(\mathbf{X}_k, \mathbf{u}_k) + \mathbf{n}_k \tag{5.4}$$

where $\mathbf{n}_k$ is Gaussian noise with zero mean and covariance $\mathbf{Q}_k$ related to the uncertainty of the velocity vector $\mathbf{u}_k$. The non-linear transition function $\mathbf{f}$ describes the state updates and it is given by:

$$\mathbf{f}(\mathbf{X}_k, \mathbf{u}_k) = \begin{bmatrix} \mathbf{x}_k \\ ax_k \\ ay_k \\ \mathbf{lan}_k^1 \\ \vdots \\ \mathbf{lan}_k^n \end{bmatrix} = \begin{bmatrix} x_k + V_k \Delta t \cos\vartheta_k + \frac{1}{2} ax_k \Delta t^2 \\ y_k + V_k \Delta t \sin\vartheta_k + \frac{1}{2} ay_k \Delta t^2 \\ \vartheta_k + \Omega_k \Delta t \\ ax_k \\ ay_k \\ lan_{x,k}^1 \\ lan_{y,k}^1 \\ \vdots \\ lan_{x,k}^n \\ lan_{y,k}^n \end{bmatrix} \tag{5.5}$$

5.3.2 Constructing the Measurement's Model

The measurement's model shows how information derived from the vehicle's sensor ties to the system's state. In this research, five sensors are considered: a GPS, an IMU, an odometer, a laser range finder and a stereo vision system. The measurements acquired by the GPS and the odometer are related to the posture of the vehicle, namely the vector $\mathbf{x}_k$ of the system's model. Thus:

$$\begin{bmatrix} \mathbf{gps}_k \\ \mathbf{odo}_k \end{bmatrix} = \begin{bmatrix} gps_{x,k} \\ gps_{y,k} \\ odo_{x,k} \\ odo_{y,k} \\ odo_{\vartheta,k} \end{bmatrix} = \begin{bmatrix} x_k \\ y_k \\ x_k \\ y_k \\ \vartheta_k \end{bmatrix} + \begin{bmatrix} \mathbf{w}_{gps} \\ \mathbf{w}_{odo} \end{bmatrix} \tag{5.6}$$

where **gps** and **odo** denote the readings from the GPS and the odometer respectively and $\mathbf{w}_{odo}, \mathbf{w}_{gps}$ are zero mean Gaussian noise associated to the odometer and GPS readings respectively.

The measurements acquired from the range sensors are the distances between the landmarks and the vehicle and can be expressed as:

$$\begin{bmatrix} \mathbf{d}^1_{1,k} \\ \vdots \\ \mathbf{d}^n_{m,k} \end{bmatrix} = \begin{bmatrix} \cos\vartheta_k & \sin\vartheta_k \\ -\sin\vartheta_k & \cos\vartheta_k \\ \vdots & \vdots \\ \cos\vartheta_k & \sin\vartheta_k \\ -\sin\vartheta_k & \cos\vartheta_k \end{bmatrix} \begin{bmatrix} lan^1_{x,k} - x_k \\ lan^1_{y,k} - y_k \\ \vdots \\ lan^n_{x,k} - x_k \\ lan^n_{y,k} - y_k \end{bmatrix} + \begin{bmatrix} \mathbf{w}^1_{1,k} \\ \vdots \\ \mathbf{w}^n_{m,k} \end{bmatrix} \tag{5.7}$$

where $\mathbf{d}^i_{j,k} = \left[dx^i_{j,k} \; dy^i_{j,k} \right]$ is the distance of the i^{th} landmark from the vehicle as measured by the j^{th} range sensor, $\mathbf{w}^i_j$ is zero mean Gaussian noise associated with the measurement of the j^{th} range sensor.

The range sensors considered in this research are a stereo vision system and a laser range finder. Thus, Equation 5.7 is modified as follows:

$$\begin{bmatrix} \mathbf{cam}_k^1 \\ \vdots \\ \mathbf{cam}_k^n \\ \mathbf{las}_k^1 \\ \vdots \\ \mathbf{las}_k^n \end{bmatrix} = \begin{bmatrix} \cos\vartheta_k & \sin\vartheta_k \\ -\sin\vartheta_k & \cos\vartheta_k \\ \vdots & \vdots \\ \cos\vartheta_k & \sin\vartheta_k \\ -\sin\vartheta_k & \cos\vartheta_k \end{bmatrix} \begin{bmatrix} lan_{x,k}^1 - x_k \\ lan_{y,k}^1 - y_k \\ \vdots \\ lan_{x,k}^n - x_k \\ lan_{y,k}^n - y_k \end{bmatrix} + \begin{bmatrix} \mathbf{w}_{cam,k}^1 \\ \vdots \\ \mathbf{w}_{cam,k}^n \\ \mathbf{w}_{las,k}^1 \\ \vdots \\ \mathbf{w}_{las,k}^n \end{bmatrix} \tag{5.8}$$

where **cam** and **las** denote the range readings from the vision systems and the laser range finder respectively.

Finally, the measurements acquired from the IMU are associated to the accelerations and the steering angle of the vehicle and can be described by the equation:

$$\left[\mathbf{imu}_k\right] = \begin{bmatrix} \cos\vartheta_k & \sin\vartheta_k & 0 \\ -\sin\vartheta_k & \cos\vartheta_k & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} ax_k \\ ay_k \\ \vartheta_k \end{bmatrix} + \mathbf{w}_{imu,k} \tag{5.9}$$

Equations 5.6, 5.8 and 5.9 are used to describe the measurement's model for the presented localization application. The measurement's model is described by the equation:

$$\mathbf{Z}_k = \mathbf{h}(\mathbf{X}_k) + \mathbf{w}_k \tag{5.10}$$

Based on sensor availability, the transition function **h** is defined as:

$$\mathbf{h}(\mathbf{X}_k) = \begin{bmatrix} \mathbf{gps}_k \\ \mathbf{odo}_k \\ \mathbf{imu}_k \\ \mathbf{cam}_k^1 \\ \mathbf{las}_k^1 \\ \vdots \end{bmatrix} \tag{5.11}$$

5.3.3 Linearization

The transition functions **f** and **h** are non linear functions. To linearize these functions, the first order Taylor expansion (Jacobian matrices of **f** and **h**) is used. The linear system's and measurement's models become:

$$\mathbf{X}_{k+1} = \mathbf{F}_k \mathbf{X}_k + \mathbf{n}_k \tag{5.12}$$

$$\mathbf{Z}_k = \mathbf{H}_k \mathbf{X}_k + \mathbf{w}_k \tag{5.13}$$

where

$$\mathbf{F}_k = \nabla \mathbf{f}(\mathbf{X}_k, \mathbf{u}_k) \tag{5.14}$$

and

$$\mathbf{H}_k = \nabla \mathbf{h}(\mathbf{X}_k) \tag{5.15}$$

5.3.4 Implementation

The EKF is implemented in two steps. In the first step (propagation) the system's current state and the error covariance matrix are estimated based on the system's model. In the second step (update) the system's state is improved by incorporating the sensor readings. Fig. 5.2 depicts the block diagram for the EKF implementation where $\mathbf{Q}_k$, $\mathbf{R}_k$ are the covariance matrices for the system's and the measurement's model respectively, and $\mathbf{P}_k$ is the error covariance.

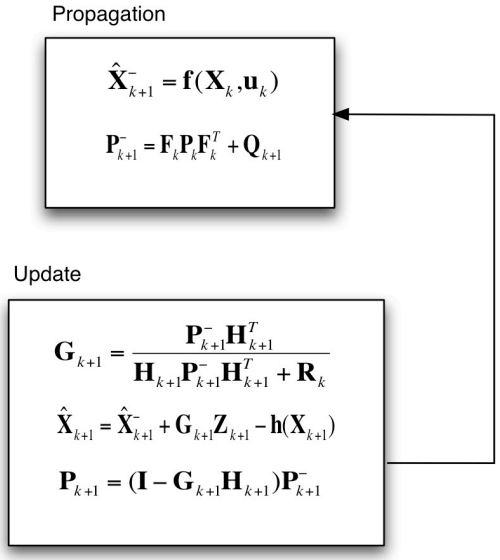


Fig. 5.2 Block diagram for the EKF implementation

5.4 Fuzzy Logic Controllers

The purpose of the EKF is to filter out the noise related to the system's and measurement's models and provide an estimate of the vehicle's position. As mentioned previously, the noise in both models is described by zero mean Gaussian distributions. However, the EKF cannot filter out the error in sensor readings unless a model of the sensor is developed. In this work, instead of creating a complex mathematical model for each sensor that will estimate the error in sensor readings, fuzzy logic has been employed to represent the error in sensor readings as zero mean Gaussian distributions. The development of the fuzzy logic controllers is based on the performance of each sensor in a series of experiments. The statistics of the error in each sensor readings are incorporate into the measurement's model through the covariance matrix $\mathbf{R}_k$.

5.4.1 Range Sensors

It has been observed that the error in range readings is related to the distance between the target and the range sensor. For example, it has been experimentally determined that the error associated with the laser readings is described by the distribution $N(0,\sigma_{\max})$ when the target's distance from the sensor is 7m to 9m, and by the distribution $N(0,\sigma_{\min})$ when the target's distance is less than 1m. Based on this observation, a Mamdani type fuzzy controller has been developed that approximates the error in range sensor readings using multiple zero mean Gaussian distributions based on the distance between the target and the sensor. In other words, the fuzzy controller computes the variance, σ, for the error distribution of every range sensor based on the sensory readings.

Three membership functions are used to describe the range measurements and the variances of the error distributions as: *Small*, *Medium* and *Large*. The FL controller for the range sensors uses the following 3 rules:

1. "If the range measurement is *Small*, then the variance is *Small*"
2. "If the range measurement is *Medium*, then the variance is *Medium*"
3. "If the range measurement is *Large*, then the variance is *Large*"

Fig. 5.3 depicts an example of the membership functions describing the distance between a range sensor and a target, and the variance of the zero mean Gaussian distribution. Detailed membership function figures for all the sensors are shown in Section 5.5.

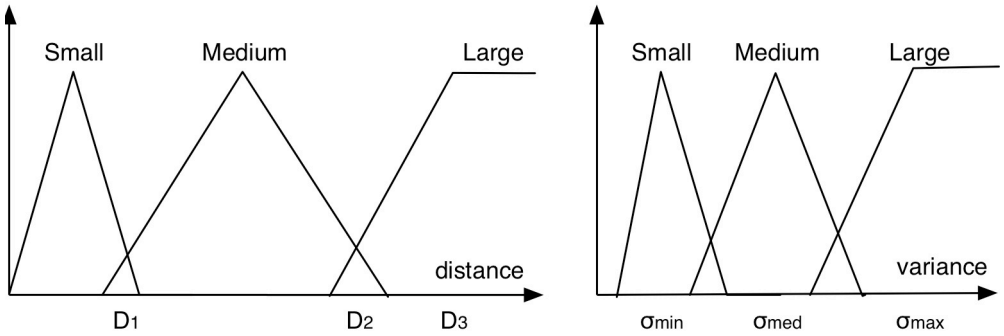


Fig. 5.3 Membership functions for the sensory readings and the variance of the error distribution

5.4.2 Odometer

The error related to the odometer readings increases as the distance the vehicle travels increases. In addition, the error in the odometer readings is closely related to the type of floor the vehicle travels on and the velocity in which the vehicle travels. For example, the error in the odometer readings is greater when the vehicle travels on tiles than when it travels on asphalt. The odometer cannot be incorporated into the EKF without an accurate error model that considers both the traveled distance and the type of floor the vehicle travels on. This is a significant drawback since the odometer data can provide an additional way of estimating the system's state.

In this work, the error in sensor readings is approximated by multiple zero mean Gaussian distributions using FL. A Mamdani type fuzzy controller has been developed that receives as inputs the type of floor the vehicle travels on and the distance that the vehicle has traveled to compute the statistics of the Gaussian distributions. The information regarding the type of floor the vehicle is traveling is controlled by the vehicle's operator.

Three membership functions have been used to describe the floor slippage, the distance traveled and the variance of the zero mean Gaussian distribution as: *Small*, *Medium* and *Large*. The FL controller designed for the odometer consists of 16 rules of the form:

1. "If the distance that the vehicle has traveled is *Small* and the floor slippage is *Small*, then the variance is *Small*"
2. "If the distance that the vehicle has traveled is *Medium* and the floor slippage is *Medium*, then the variance is *Medium*"
3. "If the distance that the vehicle has traveled is *Large* and the floor slippage is *Large*, then the variance is *Large*"

5.4.3 GPS

There are many parameters that influence the error in the GPS readings such as the satellite coverage and the weather conditions. In this work a Mamdani type FL controller has been develop that approximates the error in GPS readings with multiple zero mean Gaussian distributions based on the satellite coverage, namely the number of satellites used for triangulation. The FL controller uses three membership functions to describe the satellite coverage and the variance of the zero mean Gaussian distributions as: *Small*, *Medium* and *Large*. There are three rules in the FL controller as follows:

1. "If the coverage is *Large*, then the variance is *Small*"
2. "If the coverage is *Medium*, then the variance is *Medium*"
3. "If the coverage is *Small*, then the variance is *Large*"

5.4.4 IMU

The IMU provides information on angular rate and acceleration of the vehicle. To estimate the position of the vehicle using as sole information the acceleration of the vehicle, a double integration is required. However, the error in the IMU readings is integrated as well making the IMU useless in short time.

Unfortunately, there are no empirical rules that could be used to approximate the error in the IMU readings as zero mean Gaussian distributions. An error model needs to be developed in order to efficiently incorporate the IMU readings into the EKF. The developed error model is based on [84] and it is described in the next section.

5.4.5 Fuzzy Controllers Implementation

The FL controllers described in the previous subsections are used to update the covariance matrix $\mathbf{R}_k$ of the measurement's model in the EKF. Fig. 5.4 describes this procedure.

5.5 Case Study

The *ATRV-Jr* UGV has been used to test the performance of the fuzzy EKF. The experiments run indoors and outdoors using known color landmarks. The vision system identifies the landmarks and computes the distance between the vehicle and the land-

marks. A second set of range measurements to the landmarks is acquired from the laser range finder. A vision/laser registration algorithm is used to verify that both sensors provide range measurements for the same landmark. Additional position readings are acquired from the vehicle's odometer and GPS and acceleration readings from the IMU.

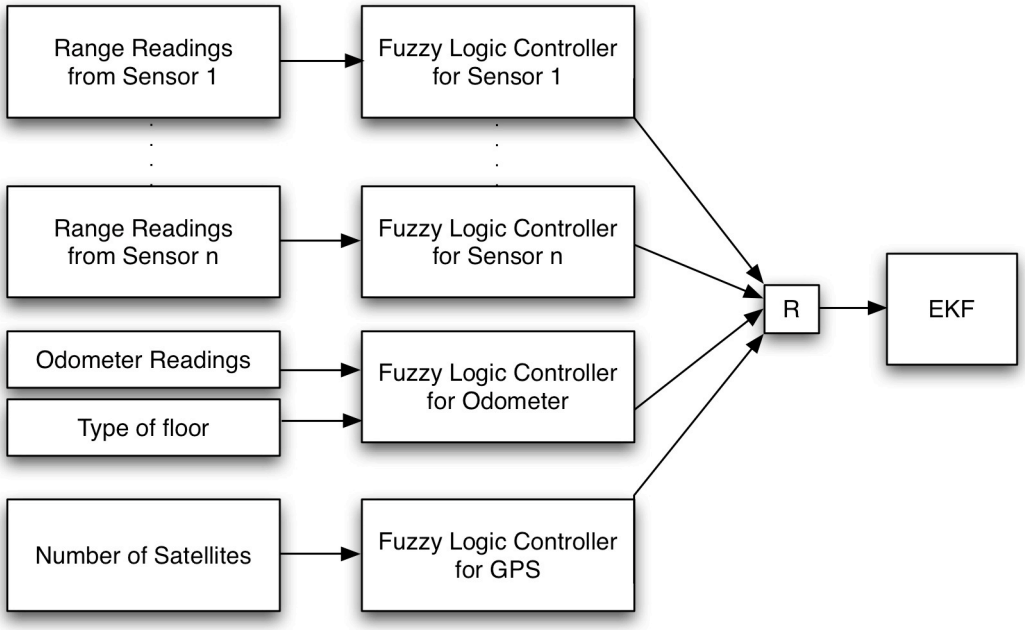


Fig. 5.4 Relation between the EKF and the fuzzy controllers

The rest of this section describes the landmark identification technique, the stereo vision distance computation technique, the vision/laser registration algorithm, the GPS data conversion, the IMU error model and the fuzzy EKF.

5.5.1 Landmark Identification and Distance Computation

The vision system of the ATRV-Jr consists of two pan/tilt color cameras mounted on top of the vehicle, at a distance of *35cm* from each other. The landmarks that have been used in these experiments have distinct color and can be separated from the image's background by a color threshold technique. The main steps of the vision algorithm that identifies a landmark and computes its distance from the vehicle are:

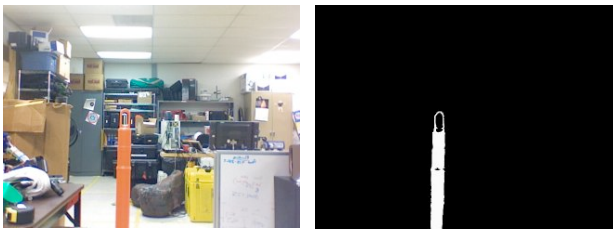
5.5.1.1 Image Acquisition

Image acquisition is achieved at a rate of *30 fps* using the Video for Linux API. Each *24bit* color image has a resolution of *320x240* pixels.

5.5.1.2 Landmark Identification

A landmark is identified by its color. Identifying color in an image instead of template matching techniques requires less computational time. The HSI color space has been used for the segmentation technique that follows. Since the landmark is a known object, the variation of the hue, saturation and intensity values of the color's representation in the HSI color space is known. A segmentation technique on *H-I* plane based on a region growing algorithm is used to separate the target from the image's background. The basic concept is to identify a pixel, "seed", in the image that takes the mean hue and intensity

values within the area of the object's color and grow a region with similar neighboring pixels. For example, the cone shown in Fig. 5.5a has hue and intensity values that vary between *(10, 20)* and *(216, 250)* respectively. Thus, the seed pixel will have a hue value of *15* and an intensity value of *233*. The region growing algorithm will merge neighboring to the seed pixels that have hue and intensity values in the former area. Fig. 5.5b shows the result of this technique.



(a) (b)

Fig. 5.5 Color threshold technique

5.5.2 Distance Computation

The method that is used to compute the distance between the vehicle and the landmark is presented in [100]. This method does not require any camera calibration technique that adds computational load to the system, but it is based on the image size, the field of view of the camera(s) and the relative position of the cameras. Fig. 5.6 demonstrates the relationship between the fields of view of the two cameras and the perspective of a world point from each camera.

The depth measurement is derived by:

$$\mathbf{cam}_k^i = \frac{320B}{2(xp_1 - xp_2)\tan(24.4)}(cm) \tag{5.16}$$

where $\mathbf{cam}_k^i$ is the distance between the left camera and the i^{th} landmark, B is the relative distance between the two cameras (baseline), $(xp_1 - xp_2)$ is the pixel disparity.

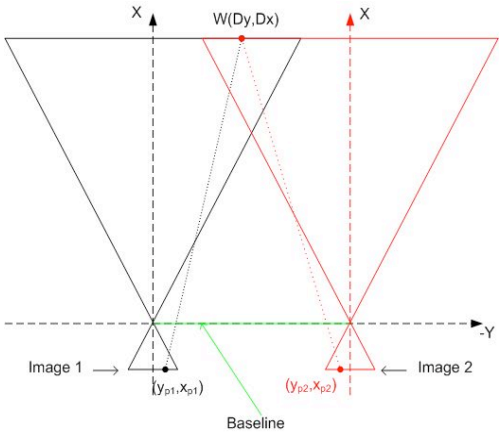


Fig. 5.6 Stereo vision system in parallel configuration

The distance between the left camera and the landmark over the Y axis is derived by:

$$cam_{y,k}^i = -\frac{2}{320} xp\mathbf{cam}_k^i \tan(24.4)(cm) \tag{5.17}$$

where xp the image location of a landmark pixel and

$$cam_{x,k}^i = \sqrt{\mathbf{cam}_k^{i\,2} - cam_{y,k}^{i\,2}} \tag{5.18}$$

5.5.2.1 Vision / Laser Registration

An essential issue on data fusion applications is that all sensors must provide data for the same entity. In this case study a stereo vision system and laser range finder sensors

mounted on top of a vehicle are used to compute distances from a landmark. Both sensors are able to recognize objects located in front of the vehicle. The purpose of the vision system/ laser registration is to verify that both sensors provide range data for the same landmark. Fig. 5.7 depicts the coordinate systems of both sensors that need to be aligned.

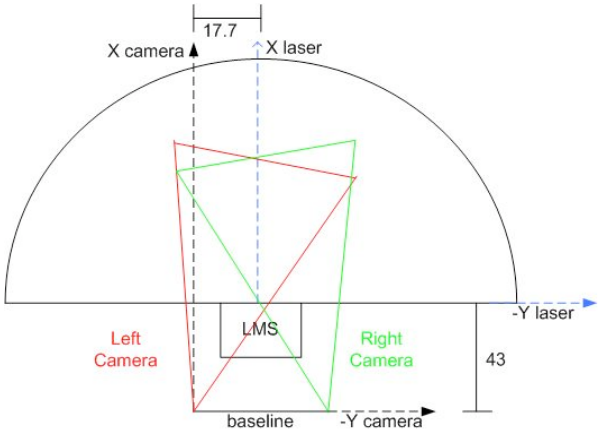


Fig. 5.7 Relation between the coordinate systems of the vision system and the laser range finder. The Fig.'s units are cm.

The depth computation from the vision system is derived from (5.16). This measurement has to be transferred to the laser's coordinate system. The transformation is achieved by translations over the X and Y axis (Fig. 5.7). Thus the distance of an object computed by the vision system in the laser's coordinate system will be:

$$las^{i}_{x,k} = cam^{i}_{x,k} - camtr_x$$

$$las^{i}_{y,k} = cam^{i}_{y,k} - camtr_y \qquad (5.19)$$

where $camtr_x = 43cm$ and $camtr_y = 17cm$

Finally, the angle in which the i^{th} lies in the laser's coordinate system is derived by:

$$\delta = arch\tan\left(\frac{las_{x,k}^{i}}{las_{y,k}^{i}}\right) \tag{5.20}$$

The range measurement from a landmark taken by the laser sensor at angle δ corresponds to the measurement taken by the vision system for the same landmark.

5.5.3 GPS Conversions

The information derived from the GPS is converted from Latitude/Longitude to Universal Transverse Mercator (UTM) coordinates. UTM is a rectilinear mapping system in map coordinates are represented as Cartesian coordinates and distance is calculated using Euclidian distance measures [101]. The units of the UTM system are meters and the equations for the conversion could be found in [101].

5.5.4 IMU

As described in [84], the error in IMU readings is approximated by the following function:

$$\varepsilon(t) = C_1(1 - e^{-\frac{t}{T}}) + C_2 \tag{5.21}$$

where C_1, C_2 and T are parameters. To estimate the values of the parameters C_1, C_2 and T, IMU readings were collected while the sensor was immobilized. Using the Levenberg-

Marquardt least square fit method, the IMU readings were fitted to Equation 5.21. Table 5.1 summarizes the best fitting parameter values for each of the IMU outputs.

Table 5.1 Values of fitting parameters

	C_1	C_2	T
Acceleration on X	-0.037 m/s^2	0.171 m/s^2	3.787 s
Acceleration on Y	2.167 m/s^2	0.016 m/s^2	735.714 s
Acceleration on Z	-0.036 m/s^2	0.178 m/s^2	1.498 s
Angle rate around X	-0.009 rad/s	0.078 rad/s	8.329 s
Angle rate around Y	-0.002 rad/s	0.002 rad/s	1.536 s
Angle rate around Z	-0.001 rad/s	0.001 rad/s	14.141 s

The IMU error model will be incorporated into the system's model of the EKF. It is assumed that the vehicle is moving in a 2D space and readings such as acceleration on Z axis, angle rates around X and Y axis are not considered. The system's model described in Equations 5.4 and 5.5 will become:

$$\mathbf{X}_{k+1} = \mathbf{f}(\mathbf{X}_k, \mathbf{u}_k) + \mathbf{n}_k \tag{5.22}$$

$$\mathbf{f}(\mathbf{X}_k, \mathbf{u}_k) = \begin{bmatrix} \mathbf{x}_k \\ \mathbf{imu}_k \\ \mathbf{lan}_k^1 \\ \vdots \\ \mathbf{lan}_k^n \end{bmatrix} = \begin{bmatrix} x_k + V_k \Delta t \cos\vartheta_k + \frac{1}{2} ax_k \Delta t^2 \\ y_k + V_k \Delta t \sin\vartheta_k + \frac{1}{2} ay_k \Delta t^2 \\ \vartheta_k + \Omega_k \Delta t \\ ax_k + eax_k \\ \frac{Tax_k}{Tax_k + \Delta t} \\ ay_k + eay_k \\ \frac{Tax_y}{Tax_y + \Delta t} \\ angle_{z,k} \\ lan_{x,k}^1 \\ lan_{y,k}^1 \\ \vdots \\ lan_{x,k}^n \\ lan_{y,k}^n \end{bmatrix} \tag{5.23}$$

where $\mathbf{imu}_k = \left[ax_k \; eax_k \; ay_k \; eay_k \; angle_{z,k} \right]^T$ the IMU readings related to the vehicle's accelerations, errors in acceleration, and steering angle at time k, and Tax_k, Tay_k the corresponding parameters T as shown in Table 5.1.

5.5.5 Fuzzy Extended Kalman Filter

This section describes in detail the FL controllers and their involvement into the EKF. A FL logic controller has been designed for each of the sensors: GPS, odometer, stereo vision system and laser range finder. These controllers are responsible for updating the statistics of the distributions describing the error in sensor readings. The design of the FL controllers involved a number of experiments that helped identify the error in sensor read-

ings in various conditions. To compute the error in odometer and GPS readings, experiments were run indoors and outdoors with the vehicle traveling at a constant speed of 0.5m/s. On the other hand, the error in range measurements was computed with the vehicle immobilized while multiple readings were collected in various distances from a target.

Table 5.2 shows the statistics of the error in the range sensors readings as a function of the measured distance and Table 5.3 summarizes the performance of the odometer in different types of floors based on the traveled distance. Table 5.4 shows the performance of the error in the GPS readings based on the number of satellites available.

Table 5.2 Distributions of the error in sensor readings with respect to the range measurements

Distance	7-9m	3-5m	0-2m
Vision System	$N(0,0.7^2)$	$N(0,0.3^2)$	$N(0,0.1^2)$
Laser	$N(0,0.6^2)$	$N(0,0.2^2)$	$N(0,0.1^2)$

Table 5.3 Distributions of the error in odometer readings with respect to the traveled distance

Distance	0-5m	10-15m	20-inf
Odometer-Tile	$N(0,0.2^2)$	$N(0,1.5^2)$	$N(0,2.5^2)$
Odometer-Grass	$N(0,0.2^2)$	$N(0,0.7^2)$	$N(0,1.6^2)$
Odometer-Asphalt	$N(0,0.2^2)$	$N(0,0.6^2)$	$N(0,1.2^2)$

Table 5.4 Distributions of the error in GPS readings with respect to the satellite coverage

Satellites	5	6	7
GPS	$N(0,7^2)$	$N(0,4^2)$	$N(0,2^2)$

Fig. 5.8 depicts the membership functions of the FL controllers designed for the stereo vision system (a), the laser range finder (b), the GPS (c) and the odometer (d). The rules for each FL controller follow the structure described in Section 5.4.

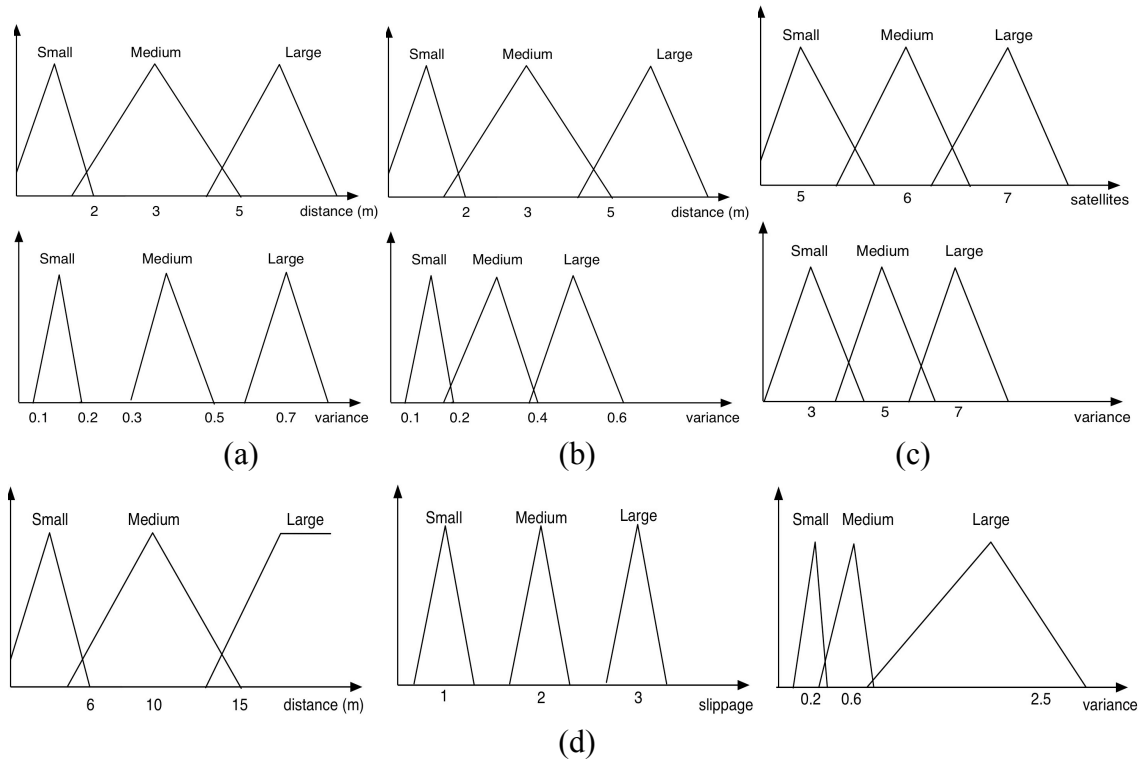


Fig. 5.8 Membership functions of the vision system (a), laser range finder (b), GPS (c) and odometer (d) FL controllers

The fuzzy EKF is implemented by recursively computing the propagation equations, the measurements covariance matrix and the update equations. Fig. 5.9 demonstrates this procedure.

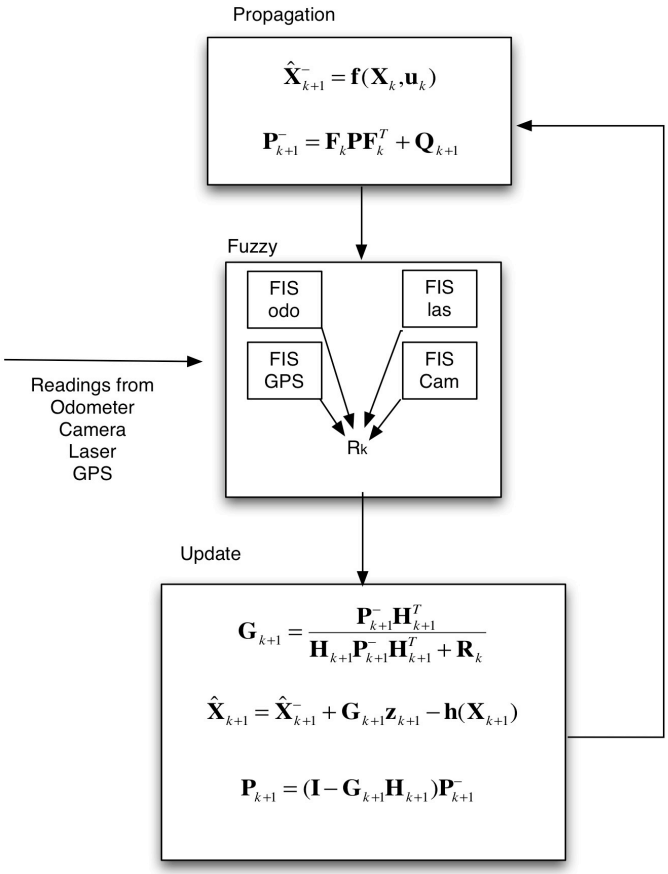


Fig. 5.9 Block diagram for the fuzzy EKF

5.5.6 Results

To evaluate the performance of the fuzzy EKF, a set of experiments was run in-doors and outdoors. The distinction between indoor and outdoor environments has to be stated because it influences considerably the sensors' performance. For example the GPS cannot establish satellite connection indoors while the IMU readings are heavily distorted due to metallic surroundings.

The performance of the fuzzy EKF is compared with the performance of the EKF. Since there are no error models for the sensors used in the EKF, the errors in sensors readings were assumed zero mean Gaussian using the maximum variance for each sensor from Tables 5.3-5.4.

5.5.6.1 Indoor Environment

For the indoor experiments the vehicle is assigned to follow two trajectories. For the first trajectory (straight line) the linear velocity is constant, *V=0.3m/s*, and the angular velocity is zero, *Ω=0rad/s*. For the second trajectory the linear velocity is constant *V=0.3m/s* and the angular velocity is repeatedly changed between *-0.2rad/s* and *0.2rad/sec*. Each cycle's time is *0.2sec*. For comparison purposes, results derived using the traditional EKF are also presented. Fig. 5.10 and 5.11 present the vehicle's trajectory as estimated by the EKF and the fuzzy EKF. For both experiments only one landmark is considered, located in front of the vehicle at a distance of 7.90m.

Table 5.5 summarizes the results for the two trajectories. As shown, for the first experiment (straight line) the error remains less than 4% for all the methods. On the other hand, for the second experiment the odometer error is more than 20%, while the position estimation error for the EKF and the fuzzy EKF is 6% and 4% respectively.

5.5.6.2 Outdoor Environment

For the outdoor experiments the vehicle was assigned to perform a square with edges of 10m. The experiments were run in different days with different weather conditions so that different satellite coverage will occur. Four landmarks are considered, one at the end of each square's edge. In the following figure, the actual position of the landmarks appears as red squares and actual vehicle position samples appear as red stars. Since the experiments were run outside for long distances, it is impossible to know the exact route the vehicle followed. The vehicle position samples and the landmark positions are as closed to the reality as possible. Fig. 5.12 and 5.13 present the performance of the EKF and the fuzzy EKF respectively. There was a lock of seven satellites for this experiment (*Large*, in terms of FL membership function). The experiments in Fig. 5.14 and 5.15 were conducted with a six satellites lock (Medium, in terms of FL membership function), while the experiments in Fig. 5.16 and 5.17 with a five satellites lock (*Small*, in terms of FL membership function).

Table 5.5 Error comparison for indoor navigation

State	Odometer		EKF		Fuzzy EKF	
	Exp. 1	Exp. 2	Exp. 1	Exp. 2	Exp. 1	Exp. 2
x	4%	5%	2%	2.3%	1%	1.3%
y	2%	25%	2%	6%	2%	4%
lan_x^1			0%	2%	0%	1%
lan_y^1			0%	5%	0%	4%

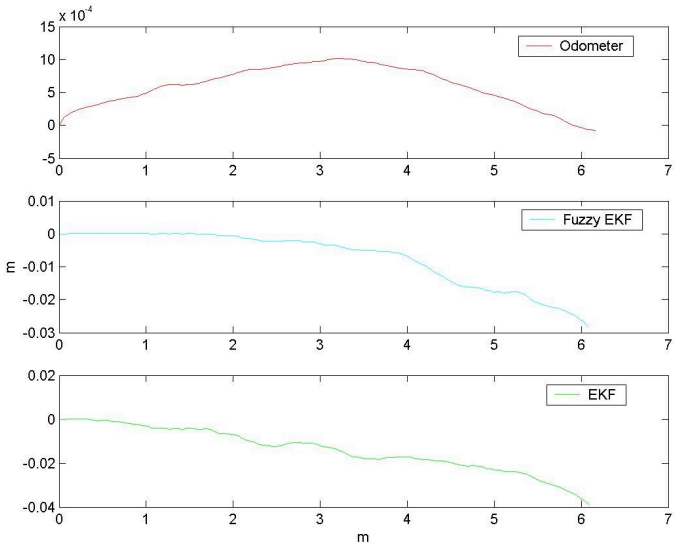


Fig. 5.10 The vehicle trajectory according to three methods, experiment 1

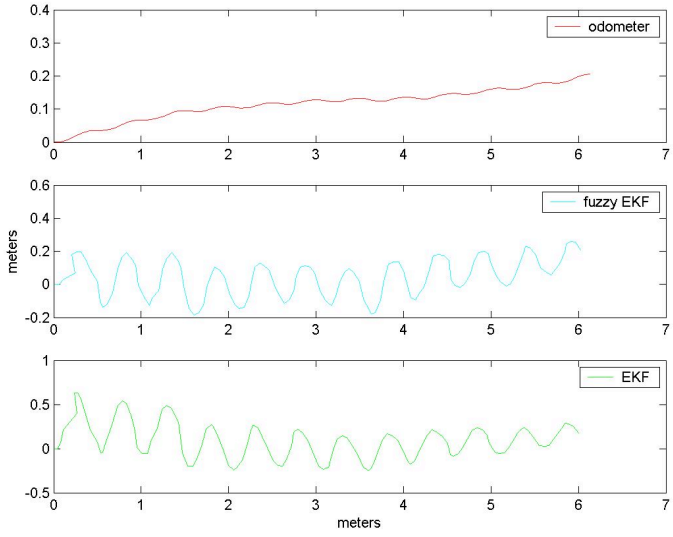


Fig. 5.11 The vehicle trajectory according to three methods, experiment 2

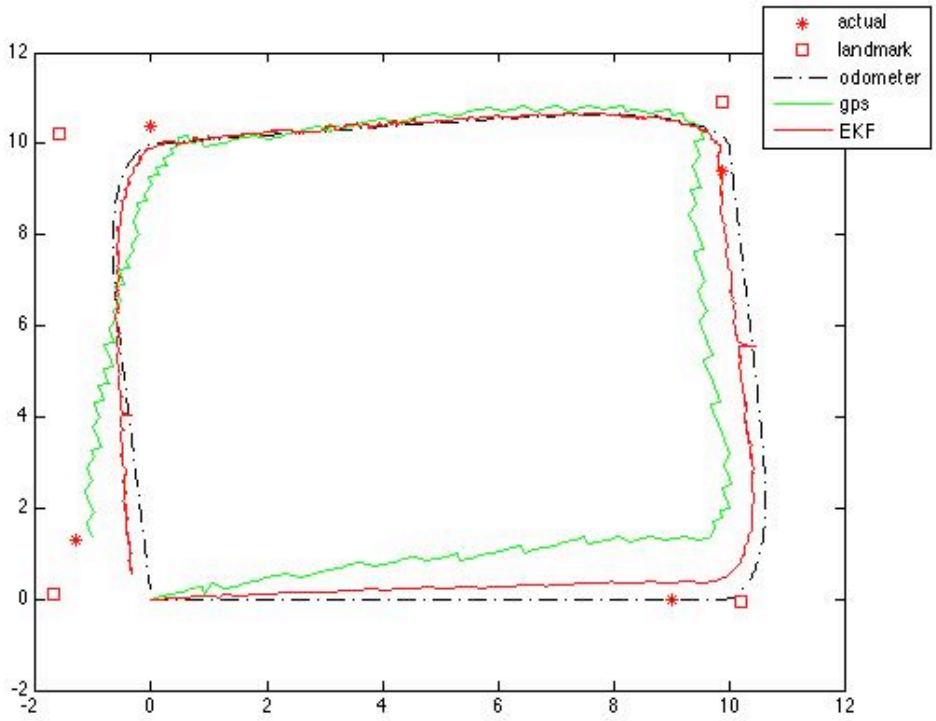


Fig. 5.12 Performance of the EKF, experiment 1

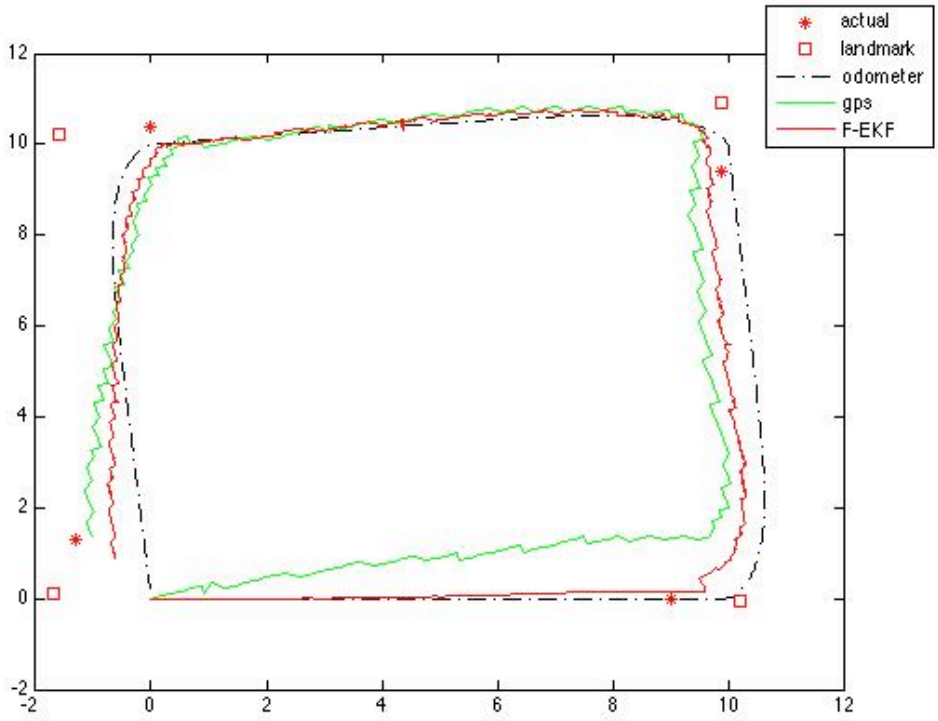


Fig. 5.13 Performance of the fuzzy EKF, experiment 1

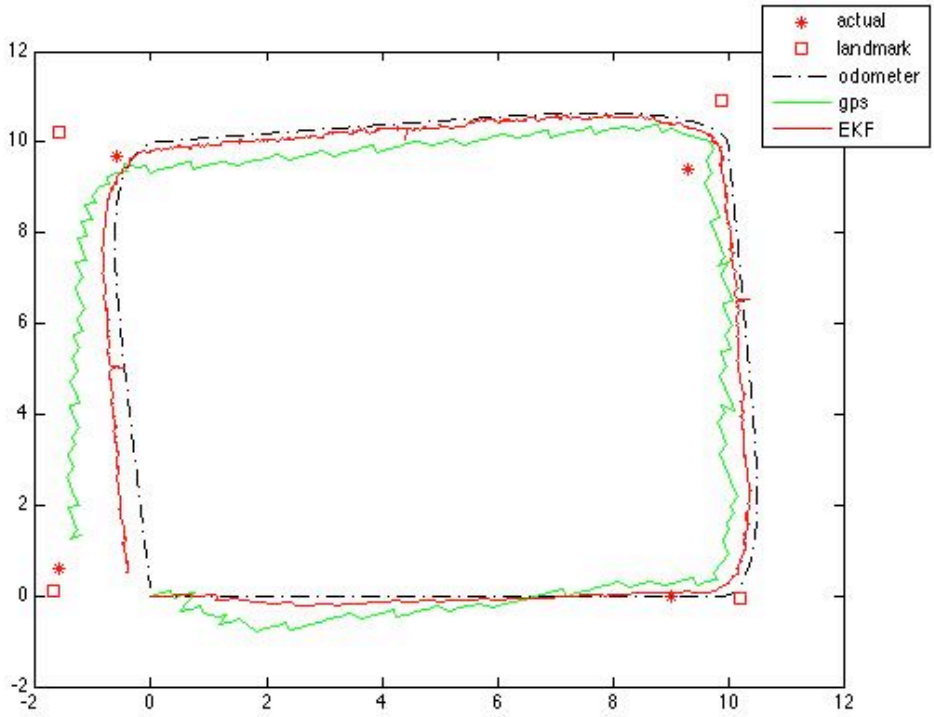


Fig. 5.14 Performance of the EKF, experiment 2

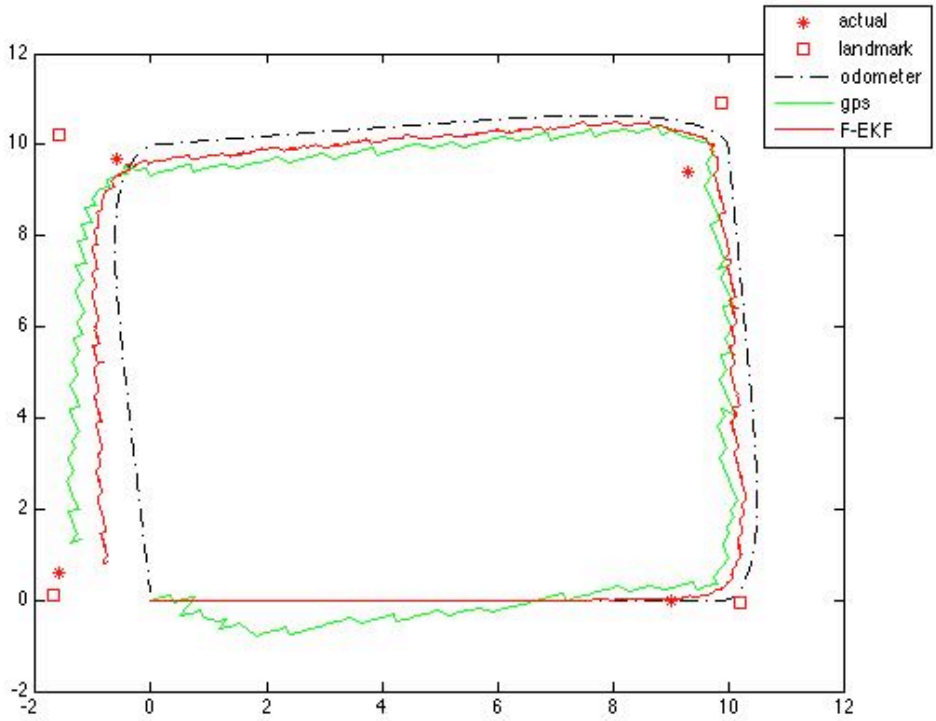


Fig. 5.15 Performance of the fuzzy EKF, experiment 2

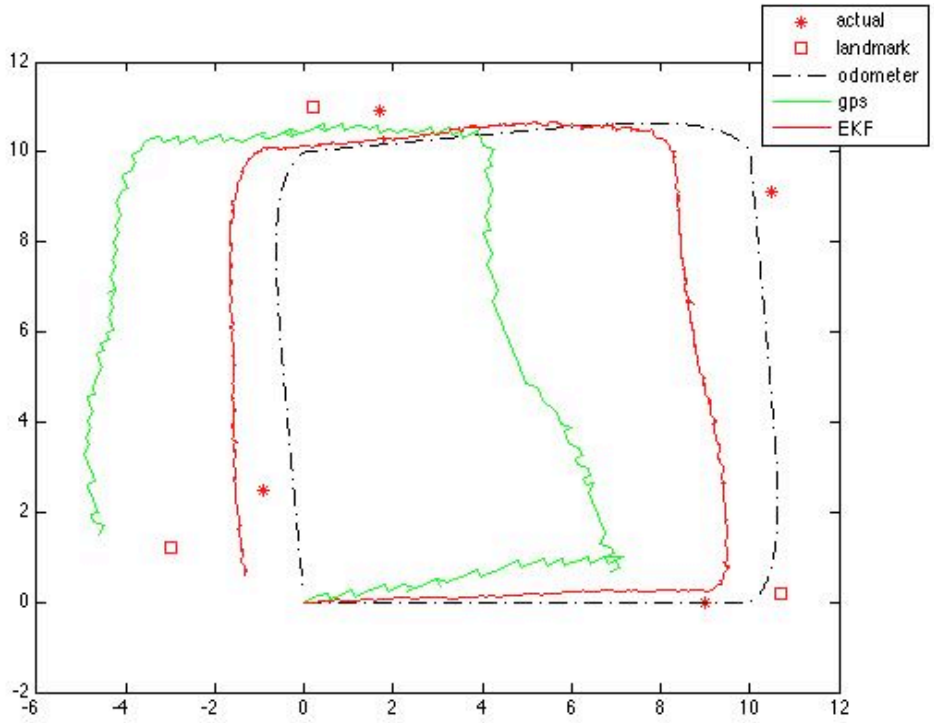


Fig. 5.16 Performance of the EKF, experiment 3

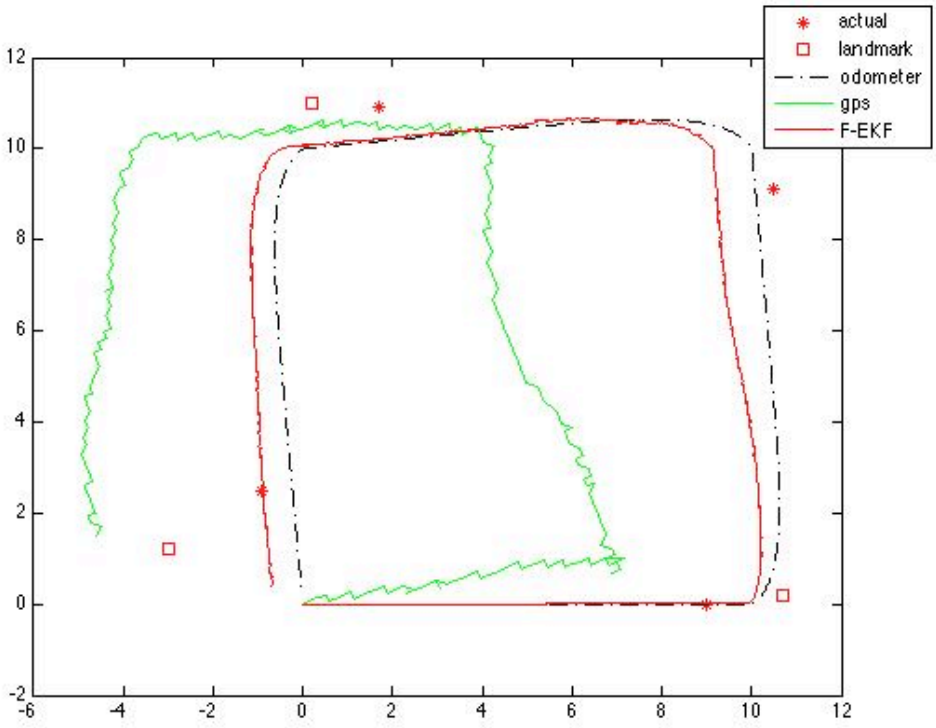


Fig. 5.17 Performance of the fuzzy EKF, experiment 3

As shown in Fig. 5.12 to 5.17 the proposed fuzzy EKF is able to fuse information from multiple sensors to estimate the position of the vehicle. In most cases, the fuzzy EKF performs better in terms of position accuracy from the EKF. This is because the fuzzy EKF takes into consideration the error in sensor readings. For example in Fig. 5.12-513 where the satellite coverage is *Large*, the fuzzy EKF considers a *Small* error in the GPS readings. Thus, the estimated route is closer to the GPS route. However, in Fig. 5.16-17 when the satellite coverage is *Small*, the fuzzy EKF considers *Large* error in the GPS readings, resulting in a route closer to the odometer's route.

5.6 Conclusions

This chapter presented a method for UGVs localization using Fuzzy Logic and Kalman Filtering. Information from five different sensors was fused to provide an estimate of the vehicles position. Fuzzy Logic has been used to compute and adjust the parameters of the error distribution of the sensor readings and update the covariance matrix of the measurement's model in EKF. As demonstrated the fuzzy EKF performs better than the EKF in terms of position accuracy. More extensive experimentation on the sensors behavior may further improve the accuracy of the fuzzy EKF.

Future work involves natural landmark selection. The vehicle should be in position to dynamically identify the most dominant landmarks of the environment that moves.

Chapter 6

Mission Planning

6.1 Introduction

Many applications in industrial, civilian and military fields benefit from mobile robot utilization. Application domains vary from warehouse patrolling to service robotics and to space exploration. Mobile robots can be assigned to explore, map or inspect friendly or hostile territories [102], [103], [104], or dispense medications in medical facilities [105]. Specifically in industrial applications, such as manufacturing, underground mining, toxic waste clean-up and material storage/handling, where many processes take place in hazardous environments harmful to human health, the choice of robotics-based solutions is justifiable. Furthermore, as the complexity and requirements of an application increases, significant advantages may be drawn from the use of multi robot systems.

Multi robot systems are classified into cooperative robot teams and robot swarms [106]. The difference between cooperative robot teams and robot swarms is that in the former case the team members present different sensory capabilities while in the latter case all members are identical. A cooperative robot team is considered in this chapter.

A major challenge when working with multi robot systems is that of task allocation and coordination. The overall mission is decomposed into multiple tasks to which one or more robots are assigned. The task allocation problem is further complicated considering the dynamic characteristics of the robot team such as robot failures and repairs that may lead to incomplete tasks. The robot team should be able to complete the mission even if some team members are no longer functional.

This chapter describes a general dynamic task allocation and controller design methodology for cooperative robot teams. The robot team is modeled as a DES where each robot is modularly represented by a finite state automaton model. The mission requirements model is synthesized from individual finite state automata representing task completion requirements. The proposed control methodology is partially based on the RW supervisory control theory [107]. However, instead of synthesizing a complete supervisor, as the traditional RW theory suggests, a limited lookahead policy is adopted that enables/disables events in the system in real-time based on the evaluation of a utility function and robot availability. The utility function uses fuzzy logic to quantify the ability of a robot to perform a task. The robot modules appear or disappear overtime depending on failure and repair events of the robots and in case of failures, the control methodology re-allocates tasks to the operational robots of the team to ensure mission completion. Our work is motivated first by the fact that in traditional supervisory control theory, the acceptable sequences of event execution determined apriori are computationally intractable for realistic size problems. Furthermore, in applications where there is a significant degree of uncertainty associated with resource reliability and the environment, these sequences may not be executable. Instead, we propose a control approach based on a lim-

ited lookahead control policy for task allocation in real time. Secondly, the criteria used in restricting system evolution are the marked states, which denote the acceptable states, and the legal language, which denotes acceptable system behavior. These criteria fail to describe a preferred behavior within the acceptable behaviors. In cooperative robot teams, several characteristics of the team members, such as endurance, reliability, efficiency etc, must be considered in task allocation decisions. We describe these characteristics as fuzzy variables and develop a fuzzy controller to determine the *utility function value* for each task allocation event. These values are then used to determine a preferred task allocation in real time as a part of the proposed controller design methodology.

To further clarify the proposed control methodology, an application scenario, depicted in Fig. 6.1, where a team of mobile robots is assigned to patrol a warehouse containing hazardous and security sensitive materials is considered throughout the chapter. Three robots with different sensory capabilities are employed. Table 6.1 summarizes the sensor suite for each robot.

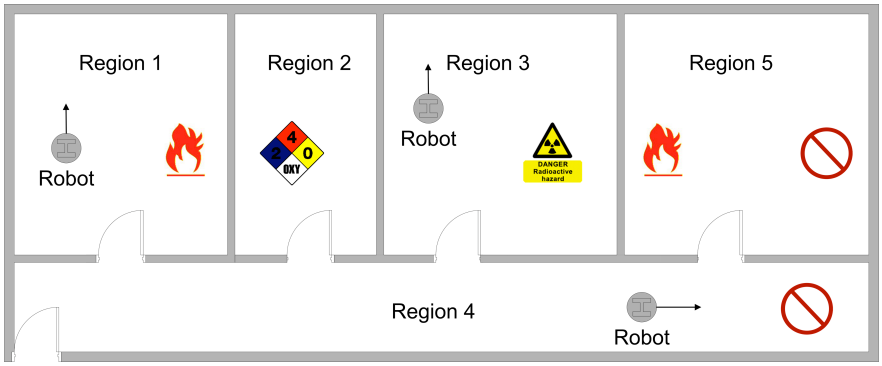


Fig. 6.1 Warehouse partitioning for the patrolling scenario

Based on the *partition based* patrolling strategy described in [108] the warehouse is partitioned into five regions as follows: Region 1 contains flammable materials, Region

2 chemical materials, Region 3 radioactive materials, Region 4 security sensitive materials and Region 5 security sensitive and flammable materials. This configuration allows us to introduce features commonly encountered in cooperative robot team missions, such as flexibility in task assignment (Region 1 can be assigned to Robot 1 or Robot 3), and cooperation between the robots (Region 5 must be assigned either to Robot 3 or first to Robot 1 and subsequently to Robot 2).

Table 6.1 Robot sensors

Robot	1	2	3
Sensors	Fire detector Chemical detector Geiger Counter	Chemical detector Geiger Counter Vision System	Fire detector Vision system

Table 6.2 Region/robot allocation

Region	1	2	3	4	5
Material	Flammable	Chemical	Radioactive	Security Sensitive	Flammable & Sec. Sensitive
Robots	Robot1 or Robot 3	Robot 1 or Robot 2	Robot 1 or Robot 2	Robot 2 or Robot 3	Robot 3 or Robots 1 & 2

The team's mission is to inspect all five warehouse regions. The overall mission is divided into 5 tasks where task $k = \{1,2,3,4,5\}$ corresponds to the patrolling/inspection of the warehouse Region *k*.

The rest of this chapter is organized as follows: Section 6.2 discusses the related literate. Section 6.3 presents the DES models of the robot team and mission requirements models for the task allocation problem. Section 6.4 describes the utility function concept

and the fuzzy controller used to determine the utility function values for task allocation events. Section 6.5 describes the proposed limited lookahead policy. Section 6.6 presents the experimental results using the proposed controller design methodology and Section 6.7 includes conclusions and future research directions.

6.2 Related Work

The task allocation problem has been addressed in literature by utility based approaches and auction based approaches for both cooperative robot teams and robot swarms. Utility based approaches have been used for task allocation in many control architectures as in [106], [109] and [110]. Each task is assigned to a robot based on various utility estimates: In [106] each robot is assigned a task based on utility estimates of *acquiescence* and *impatience.* In [109] utilities are computed as a function of relevant sensors; the robot having the most relevant sensors for a task is assigned the particular task. Utility has also been used in robot team cooperation to estimate the cost of executing an action [111] and for sensor-based metrics [112]. Auction based approaches as in [113], [111] and [112] achieve task allocation based on the Artificial Intelligence concept of Contract Net Protocol [114]. Each robot *bids* for an available task and the robot with the higher bid is assigned to that task. In the proposed control methodology, the dynamic task allocation problem is addressed using utility and fuzzy logic. Utility function values are computed based on the ability of each robot to perform a task considering several factors.

Limited lookahead policies for supervisory control have been first studied in [115] where a limited lookahead window is used to control the online behavior of the uncontrolled system model. The notion of *pending traces* is introduced to describe the legality of a trace in the lookahead window based on a conservative or an optimistic attitude. The notion of pending traces was later raised in [116] by extending the uncontrolled system model behavior by arbitrary traces beyond the limited lookahead window. In [117], the authors present a methodology that recursively computes the future control actions based on previously computed control actions. Later, in [118] and in [119] the authors present an extension to the lookahead policies to cope with the computational complexity problem by making a control decision without exploring the whole lookahead window.

Further enhancements in limited lookahead policies for supervisory control have been proposed. In [120] a lookahead policy is presented for systems with partial observability. Also, in [121] system's uncertainty is considered by assigning probabilities to event occurrences and in [122] by modeling all possible variations of the system. To our knowledge there are no limited lookahead policies in the literature designed to control cooperative robot teams.

As noted in [123], only few approaches, as described in [124] and [125], concentrate in time varying systems where system modules appear or disappear in time. In these approaches resource modules disappear only after the completion of assigned tasks. In this work, we relax this assumption by considering failures during task execution. In coordinated robot teams the concept of robot failures and repairs is important since a robot failure while executing a task will lead to an incomplete mission unless the control model

reassigns the task elsewhere. The lookahead policy presented in this chapter considers robot failures and repairs to ensure mission completion.

Supervisory control based approaches on discrete event system have been used by a number of researches to control mobile robot teams. However, although a limited amount of work considers robot failures, not much effort is found in the area of control decisions concerning robot rejoining the robot team after repairs. Specifically, the automata based approaches presented in [126], [127] and [128] consider situations where some robots go offline but do not take into account situations where robots come back online. Similarly, the Petri Net controller in [129] disregards robot repairs. Finally, the control architecture presented in [130] handles only robot failures.

6.3 System Model Description

In RW supervisory control theory, the uncontrolled system's model (UCSM) and the mission requirements are separately modeled using finite automata. Considering the patrolling application described in Section 6.1, let $G_j = (\Sigma_j, Q_j, q_{j0}, \delta_j, Q_{jm})$ be the finite automaton (FA) representing the uncontrollable behavior of Robot j. Σ_j is the set of events Robot $j = \{1,2,3\}$ can execute, Q_j is the set of states and $\delta_j : \Sigma_j \times Q_j \to Q_j$ is the transition function. q_{j0} and Q_{jm} are the initial and final states respectively. The set of events Σ_j consists of the controllable and the uncontrollable events $\Sigma_j = \Sigma_{jc} \cup \Sigma_{ju}$ and $\Sigma_{jc} \cap \Sigma_{ju} = \varnothing$.

For the patrolling application, the controllable events are $\Sigma_{jc} = \{start_{jk}\}$ corresponding to "initiation of Task *k* by Robot *j*". Based on the robot sensory capabilities described in Tables 6.1 and 6.2, possible robot-task allocations for Task k and Robot j are defined as:

$$k = \begin{cases} \{1,2,3,5\} & ,if \quad j = 1 \\ \{2,3,4,5\} & ,if \quad j = 2 \\ \{1,4,5\} & ,if \quad j = 3 \end{cases} \tag{6.1}$$

The uncontrollable events are $\Sigma_{ju} = \{complete_{jk}, failure_{jk}, repair_{jk}, drop_{jk}\}$ corresponding to "completion of Task *k* by Robot *j*", "failure of Robot *j* while executing Task *k*", "repair of Robot *j*" and "repair of Robot *j* and re-initialization of Task *k* " respectively. The set of states for robot *j* is defined as $Q_j = \{I_j, B_{jk}, F_{jk}\}$ where I_j denotes the state of Robot j as idle, B_{jk} as busy with Task *k* and F_{jk} as failed while executing Task *k*. The initial and final states of the Robot *j* are idle, namely $q_{j0} = Q_{jm} = \{I_j\}$. Fig. 6.2, depicts the transition graph for the automaton describing Robot *j*. An arrow marks the initial state. The marked state is shown as a dark circle.

The FA model design incorporates two different cases of robot failures and repairs. A robot failure may be considered as temporary failure or failure with task reinitialization. In the first case, the failed robot will continue executing its task as soon as is repaired while in the second case the task of the failed robot needs to be reinitialized.

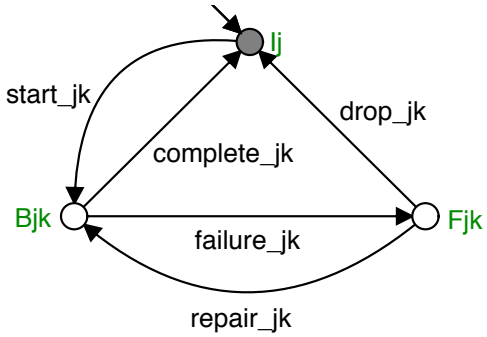


Fig. 6.2 Transition graph for Robot j

The UCSM that represents the uncontrolled behavior of the robot team is composed of the synchronous product [131] of the individual robot modules as follows:

$$G = (\Sigma, Q, q_0, \delta, Q_m) = G_1 \,||\, G_2 \,||\, G_3 \tag{6.2}$$

The specification's model is the finite automaton that models the mission requirements which is synthesized using individual task completion requirement models. Three alternative task completion requirements are modeled:

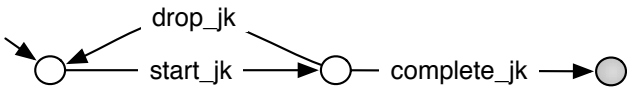
1. Alternative 1: Task k can be performed only by Robot j
2. Alternative 2: Flexibility in task assignments: Task k can be performed by Robot j or by Robot $i, i \neq j$
3. Alternative 3: Task sequencing and robot coordination: Task k must be performed first by Robot j and subsequently by Robot i, $i \neq j$

Fig. 6.3 depicts the transition graphs for these three alternative task completion requirements. Fig. 6.3a describes the requirement where Task *k* can be performed only by Robot *j*. If Robot *j* fails during the task, the task may be re-initialized as shown by the $drop_{jk}$ event. Fig. 6.3b describes flexibility in task assignment where Task *k* can be per-

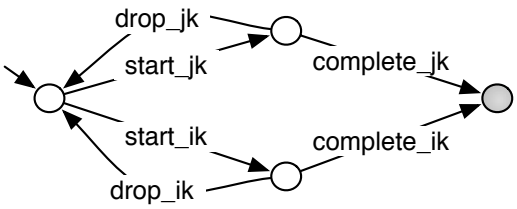
formed only by Robot *j* or Robot *i.* Finally, Fig. 6.3c describes task sequencing and robot coordination where Task *j* must be completed first by using Robot *j* and then Robot *i*. For example, consider the patrolling scenario where Robots 1 and 2 must perform Task 5 (patrol of region 5 of the warehouse). The region contains flammable and security sensitive materials. Since Robot 2 is not equipped with fire detection sensors (Table 6.1) it is vulnerable to fire. For this reason Task 5 must be assigned first to Robot 1 and then to Robot 2. The task completion requirements are modeled as finite automata of the form:

$$R_n = (\Sigma_n, X_n, \xi_n, x_{n0}, X_{nm}) \tag{6.3}$$

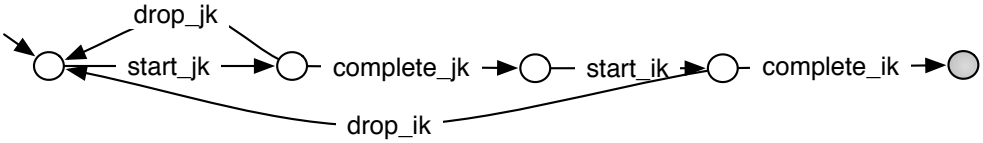
where n is the number of the individual task completion requirements. For the patrolling scenario described in Section 6.1, $n = 5$ corresponding to patrolling of the five warehouse regions. Patrol warehouse Regions 1-4 are modeled based on Alternative 2 and warehouse Region 5 is modeled based on Alternative 3.



a. Alternative 1



b. Alternative 2



c. Alternative 3

Fig. 6.3 Transition graphs for the task completion requirements

For the particular warehouse patrolling application, it is assumed that only one robot may fail once during the mission. Otherwise, it may not possible to complete the mission and further discussion of the control methodology is rendered impossible. This assumption is modeled as a finite automaton $R = (\Sigma_R, X_R, \xi_R, x_{R0}, X_{Rm})$ and is depicted in Fig. 6.4.

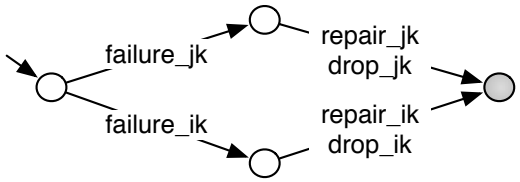


Fig. 6.4 Transition graph of the failure assumption automaton

The specification's model, S, is a finite automaton synthesized by the synchronous product of all mission requirements. Considering the specifications outlined in the warehouse patrolling scenario, the specifications model is synthesized as follows:

$$S = (\Sigma, X, \xi, x_0, X_m) = R_1 \,||\, ... \,||\, R_n \,||\, R \tag{6.4}$$

The supervisory control model for the team of robots consists of the coupled system model S/G and the control pattern Ψ. The coupled model is defined as the product of the UCSM and the specifications model, which includes all the events that are allowed by both models:

$$S/G = (\Sigma_S \cap \Sigma_G, X \times Q, \gamma = \xi \times \delta, (q_{0S}, q_{0G}), X_m \times Q_m) \tag{6.5}$$

The control pattern Ψ is a function $\Psi : \Sigma \times X \to (0,1)$ based on the supremal-controllable language of the coupled model that enables (1) or disables (0) the controllable events in the UCSM so that desirable system behavior is guaranteed. The synthesis of the control pattern and consequently the solution to the supervisory control problem is a

computationally prohibitive procedure for larger systems. Furthermore, while a task allocation and desired control pattern determined a priori may be executable with reliable resources in controlled environments, such a sequence of events is very unlikely to be executed to completion in applications associated with cooperative robot teams due to unreliable resources and uncontrollable, and possibly hostile, environments which robot teams typically operate in.

In this chapter, instead of following the traditional supervisory control approach and synthesizing the complete supervisor for the system, a limited lookahead control policy is adopted. The limited lookahead control approaches are suitable for highly dynamic systems since only a portion of the system corresponding to the system's behavior in the near future is evaluated. A limited lookahead window of finite depth is used to direct the behavior of the system. Every time an event is executed in the UCSM, the lookahead window is reconstructed and all possible sequences of events in the lookahead window are evaluated. The event leading to the highest evaluated string is enabled while the rest of the controllable events are disabled. The evaluation criteria based on the utility concept are described in the next section.

6.4 Utility Function Definition

In cooperative robot teams, each robot possesses unique characteristics including but not limited to sensory capabilities, cost, efficiency and endurance. For this reason, each robot presents a different level of ability to perform a certain task. In addition, the

utility function may be used to capture the system designer's choice to assign certain tasks to specific robots. For example, consider the case where the system's designer knows that a sensor in a robot is functional but not efficient. For this reason, the designer wishes to assign a task to this robot that does not make use of the particular sensor. These aspects complicate the task allocation problem.

An evaluation method that maximizes the overall performance of the robot team is required. In supervisory control theory, traditional system evaluation criteria are the marked states, which denote the acceptable states, and the legal language, which denotes acceptable system behavior. However, these criteria fail to describe undesirable yet acceptable behavior. For example, Robots 1 or 3 can perform Task 1. Assigning the task to either robot is an acceptable action but assigning Task 1 to Robot 1 ties up other sensory capabilities. In a sense, Robot 1 is "over qualified" to perform Task 1, making Robot 3 the more desirable choice.

The proposed control methodology employs a *utility function* that evaluates strings in the lookahead window. We define a utility function $u : \Sigma \to [0,1]$, which associates an event σ_{jk} in Σ with a utility value between 0 and 1 and we define the utility of a string s as:

$$U(s) = \sum_{\sigma_{jk} \in s} u(\sigma_{jk}). \tag{6.6}$$

The attributes mentioned that could be used to compute the robot's ability to perform a task, such as endurance, efficiency and designer's choice, represent vague concepts hard to describe mathematically. However, these concepts can be described in terms of fuzzy logic as fuzzy variables with linguistic membership functions. A Mamdani type

fuzzy logic controller [132] is proposed that receives as inputs the membership of each fuzzy variable and computes the ability of a robot to perform a task.

Three fuzzy variables are considered: robot's endurance, designer's choice and robot's efficiency. The first fuzzy variable has three membership functions {*short*, *fair*, *long*} and denotes how long a robot can remain functional. The second fuzzy variable with three membership functions {*low*, *medium*, *high*} denotes the system designer's choice to assign certain tasks to specific robots. Finally, the third fuzzy variable with three membership functions {*low*, *medium*, *high*} denotes the robot's efficiency level. The output of the fuzzy logic controller is also a fuzzy variable with membership functions {*low*, *medium*, *high*} denoting a robots ability to perform a task. Fig. 6.5 and 6.6 depict the fuzzy variables and their membership functions we have adopted for the patrolling application.

Considering a Task k and a Robot j, the ability of Robot j to perform Task k, denoted by $ability_{jk}$, is computed based on a set of rules such as:

1. If the robot's endurance is *long*, the designer's choice is *high* and the robot's efficiency is *high*, then the $ability_{jk}$ of Robot j to perform Task k is *high*
2. If the robot's endurance is *fair*, the designer's choice is *medium* and the robot's efficiency is *medium*, then the $ability_{jk}$ of Robot j to perform Task k is *medium*
3. If the robot's endurance is *short*, the designer's choice is *low* and the robot's efficiency is *low*, then the $ability_{jk}$ of Robot j to perform the task k is *low*

There are 27 such rules in the fuzzy controller, which cover all combinations among the membership functions of the input variables.

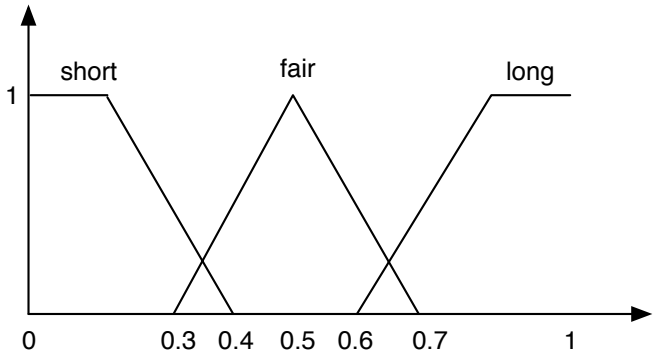


Fig. 6.5 Membership functions of the fuzzy variable: robot's endurance

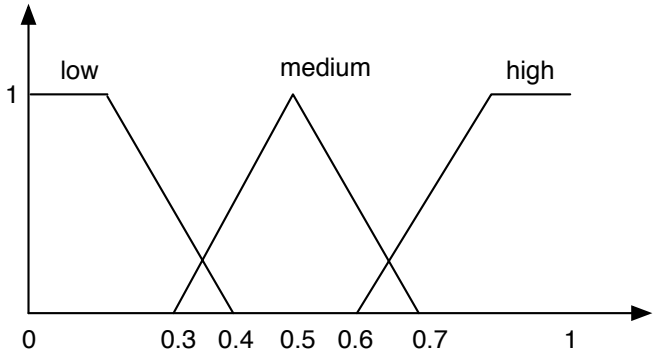


Fig. 6.6 Membership functions of the fuzzy variables: designer's choice, robot's efficiency and robot's ability

The ability of a robot to perform a task is closely related to task allocation and consequently to task initiation events $\{start_{jk}\}$. Thus, the utility function value of the events $\{start_{jk}\}$ is equal to the ability of Robot j to perform Task k. In other words,

$$u(\sigma) = ability_{jk} \text{ where } \sigma \in \{start_{jk}\}. \tag{6.7}$$

The events $\{drop_{jk}\}$ corresponding to task re-initialization due to robot failure represent cancelation of task assignments. The utility function values of the $\{drop_{jk}\}$ events are:

$$u(\sigma) = -ability_{jk} \text{ where } \sigma \in \{drop_{jk}\}. \tag{6.8}$$

The higher the utility function value for an event, the more desired this event is. Since uncontrollable events $\{complete_{jk},\ failure_{jk},\ repair_{jk}\}$ cannot be enabled/disabled, their utility function values are

$$u(\sigma) = 0 \text{ where } \sigma \in \{complete_{jk},\ failure_{jk},\ repair_{jk}\}. \tag{6.9}$$

Each time an event σ occurs in the UCSM and the limited lookahead window is reconstructed, the utility function values for all the strings $s \in L(S/G)$ in the lookahead window are computed. The string with the highest utility function value corresponds to the most desirable system behavior.

The maximization procedure is implemented as a dynamic programming problem [133] with a forward sweep and a backtracking pass. Fig. 6.7a shows a portion of the coupled model for the patrolling application. For example, assume that:

$$\begin{aligned} &u(start_{11}) = 0.5 \\ &u(start_{12}) = 1 \\ &u(start_{22}) = 0.5 \\ &u(start_{23}) = 1 \end{aligned} \tag{6.10}$$

In the forward sweep of the dynamic programming, each state Q in the lookahead tree is assigned the maximum utility function, denoted by V, of the strings that led to that state:

$$V(Q) = \max\{U(s_1), U(s_2), ..., U(s_n)\} \tag{6.11}$$

where $s_z, z = 1,2,...,n$ denotes all the strings from the current root state, q_c, to state Q, or formally $\gamma(s_z, q_c) = Q$ where γ is the transition function for the coupled model.

In Fig. 6.7a, using Equations (6.6) to (6.11), state S1 is assigned the value:

$$V(S1) = U(start_{11}) = u(start_{11}) = 0.5 \tag{6.12}$$

Similarly, the value of state S4 is the utility function of the string $start_{11}start_{22}$:

$$V(S4) = U(start_{11}start_{22}) = u(start_{11}) + u(start_{22}) = 1 \tag{6.13}$$

In the same manner, the values of states S3, S5 and S6 are:

$$V(S3) = U(start_{11}complete_{11}) = u(start_{11}) + u(complete_{11}) = 0.5 \tag{6.14}$$

$$V(S5) = U(start_{12}start_{23}) = u(start_{12}) + u(start_{23}) = 2 \tag{6.15}$$

$$V(S6) = U(start_{12}complete_{12}) = u(start_{12}) + u(complete_{12}) = 1 \tag{6.16}$$

The backtracking pass initiated after the conclusion of the forward sweep is illustrated in Fig. 6.7b. Starting from the final states in the lookahead window, a maximum utility value, V, is assigned to all of the immediate predecessors of the final states based on the equation:

$$V(Q_i) = \max\{V(Q_{iz})\} \tag{6.17}$$

where Q_i denotes a state in the lookahead window and Q_{iz} denotes the set of immediate successors of Q_i that produced the successor with the maximum utility value where:

$$\gamma(\sigma, Q_i) = Q_{iz} \text{ and } U(Q_i) + u(\sigma) = V(Q_{iz}). \tag{6.18}$$

The procedure is repeated until the root state of the lookahead window is reached. In the example of Fig. 6.7 the string with the highest value of V is $start_{12}start_{23}$. Therefore, the control methodology disables the event $start_{11}$ and enables the event $start_{12}$. A

new limited lookahead window is constructed as soon as an event is executed and the utility maximization procedure is repeated.

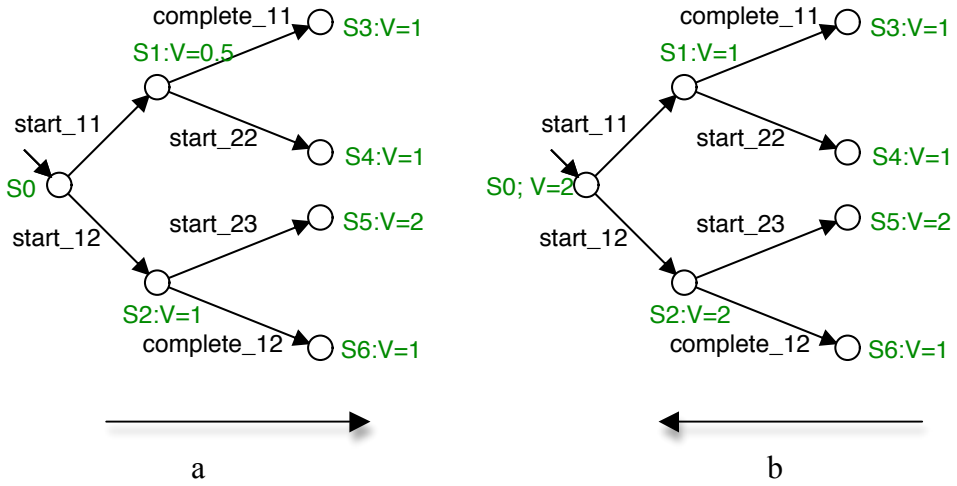


Fig. 6.7 Forward sweep (a) and backtracking pass (b) of dynamic programming

The assignment of utility function values to events is a design parameter, which is customizable based on the goals of the mission and the designer. In the task allocation scenario we are considering, these values are assigned in a manner where strings completing a task assignment are not highly evaluated as shown in Equations (6.14) and (6.16). However, a different strategy may require prioritizing task completion. In this case events associated with completion events may be assigned higher utility function values and the controller will evaluate strings with task completion events highly and guide the system accordingly.

6.4.1 Normalized Utility Function

In the evaluation process described in the previous section, a string that includes many task initiation events is evaluated higher than a string with fewer task initiation events. However, the highest evaluated string may not always correspond to the most desirable task allocation. Consider the case where $u(start_{11}) = u(start_{22}) = u(start_{34}) = 1$ and $u(start_{31}) = 2$ indicating that assigning Task 1 to Robot 3 is the desired action since $u(start_{31}) > u(start_{11})$. Suppose that the event $start_{11}$ (initiation of Task 1 from Robot 1) is a part of the string $start_{11}start_{22}start_{34}$ and the event $start_{31}$ (initiation of Task 1 from Robot 3) is a part of the string $start_{31}complete_{31}complete_{jk}$ in a limited lookahead window with depth 3 where only 3 future events are considered. The utility function value for the string $start_{11}start_{22}start_{34}$ is higher than the utility of the string $start_{31}complete_{31}complete_{jk}$ (3 and 2 respectively). Thus, the control methodology will disable the event $start_{31}$, which is a more desirable task allocation. To eliminate this bias arising from the limited depth of the lookahead window, a normalized utility function

$$U_N(s) = \frac{1}{N} \sum_{\sigma_{jk} \in s} u(\sigma_{jk}) \tag{6.19}$$

where N is the number of task initiation events in the string s is used. The normalized utility values of the two strings would be 1 and 2 respectively leading to the preferred choice of task allocation. It should be noted that the normalization of the utility function is also a design consideration and is customizable depending on the characteristics of the mission.

6.5 Limited Lookahead Control Policy

This section presents the limited lookahead policy of the proposed control methodology for task allocation in a mobile robot team. Fig. 6.8 depicts the block diagram of the control methodology algorithm consisting of 4 main modules: system initiation, lookahead window formation, string evaluation and control decision.

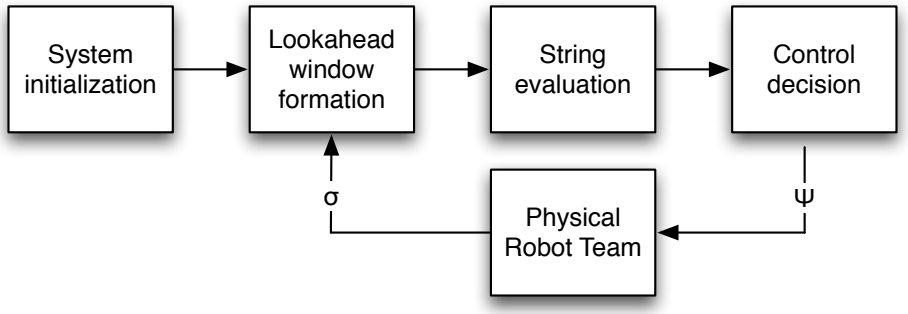


Fig. 6.8 Block diagram of the control algorithm

In the system initialization module, the UCSM and the specifications are generated as described in Section 6.2. Based on the fuzzy logic controller, each event is assigned a utility function value using Equations (6.7), (6.8) and (6.9).

Using as root state the initial state of the UCSM and the specifications model, a lookahead window is formed that includes all the transitions starting from the initial state up to a certain predefined depth in the coupled model. The transitions in the lookahead window form a tree of strings that can be executed in the UCSM. Each string in the lookahead window is evaluated using the normalized utility function shown in Equation (6.19).

In the control decision module, the event exiting the root state leading the string with the highest utility function value is enabled. All the other controllable events exiting the root state are disabled. In the case that two or more strings present the same utility function value all events leading these strings are enabled. At the next state, the new system state becomes the root state for the lookahead window and the procedure is repeated until the system reaches a marked state.

For the warehouse patrolling application, the forward sweep of the dynamic programming for a small section of the lookahead window of depth 3 is shown in Fig. 6.9a. The task initiation events $\{start_{jk}\}$ are labeled as $\{s\_jk\}$ and the task completion events $\{complete_{jk}\}$ as $\{c\_jk\}$. The utility function value for each event, u, is shown underneath the event label and the maximum utility function, V(Q), computed using Equation (6.11) is depicted at the final state of each string. Fig. 6.9b shows the result of the backtracking pass where each state has been assigned the maximum utility function based on Equation (6.17). The string with the highest utility function value appears in bold. Since event $start_{35}$ is the leading event of the string with the highest utility function value, $start_{35}$ remains enabled while all other controllable events exiting the root state are disabled.

6.5.1 Robot Failures and Repairs

Frequently, during a mission, a robot may go offline due to a sensor failure or communication loss resulting in an incomplete task. The control methodology should be able to compensate for robot failures by reallocating tasks to the operational robots of the

team. Two kinds of failures are considered in this work: temporary failures and failures with task re-initialization.

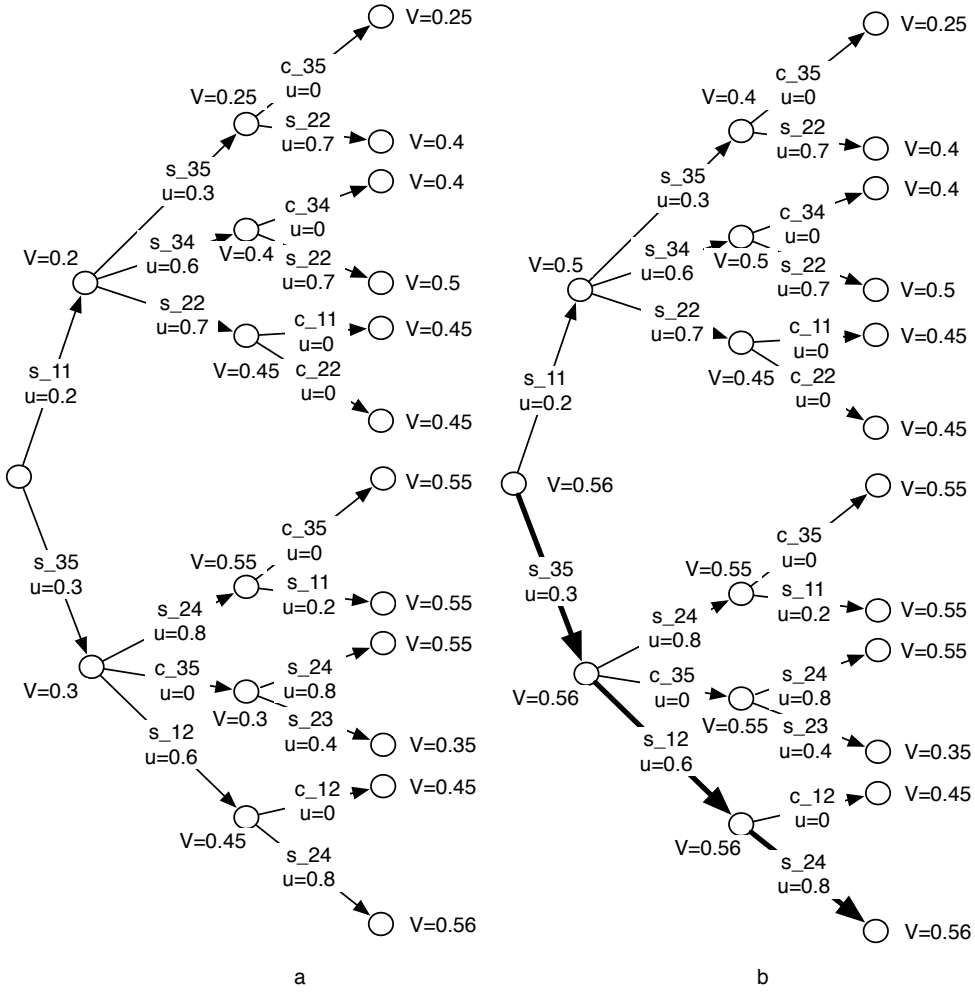


Fig. 6.9 Limited lookahead policy for the patrolling scenario

Temporary failures, such as communication loss are failures that can be repaired in a short period of time and the robot may continue executing its task after the repair. When a temporary failure occurs the events in the lookahead window associated with

task initiation or completion of the failed robot are masked until the robot is repaired so that events associated with the failed robot are not considered. Masking is used to hide all the events associated with the failed robot. In essence, since the failed robot cease to be a part of the robot team until it is repaired, its FA module disappears from the UCSM. When the robot is repaired and its FA module appears into the UCSM, all events associated to the repaired robot are unmasked.

To incorporate the failure information into the control methodology an additional module called *failure detection* has been implemented. The module interacts with the rest of the algorithm components as shown in Fig. 6.10. When the failure detection module detects a robot failure in the robot team, the information to be used in the calculation of the new limited lookahead window is forwarded to the next module. This information includes the set of events to be masked, the type of failure, and the expected time to repair.

Fig. 6.11 demonstrates a temporary failure. Robot 1 is failed in State 2 and repaired in State 5 to continue executing its task assignment. The initiation events $\{start_{jk}\}$ are labeled as $\{s\_jk\}$, the completion events $\{complete_{jk}\}$ as $\{c\_jk\}$, the failure events $\{failure_{jk}\}$ as $\{f\_jk\}$, the repair events $\{repair_{jk}\}$as $\{r_{jk}\}$and the task drop events $\{drop_{jk}\}$ as $\{dr\_jk\}$.

Failures with task re-initialization are failures that require re-initialization of the task assigned to the failed robot. For example, consider a sensor failure that it is not immediately recognized. However, during task execution the sensor failure is realized and the task needs to be re-initialized. When a temporary failure with task re-initialization

event occurs, all initiation and completion events in the lookahead window associated with the failed robot are masked until the robot is repaired. The task, assigned to the failed robot, is re-allocated to a different robot. Fig. 6.12 demonstrates a temporary failure with task re-initialization. Robot 3 has failed in State 3 while executing Task 1 and the failure is considered as failure with task re-initialization. In State 7 Task 1 is dropped and re-assigned to Robot 1.

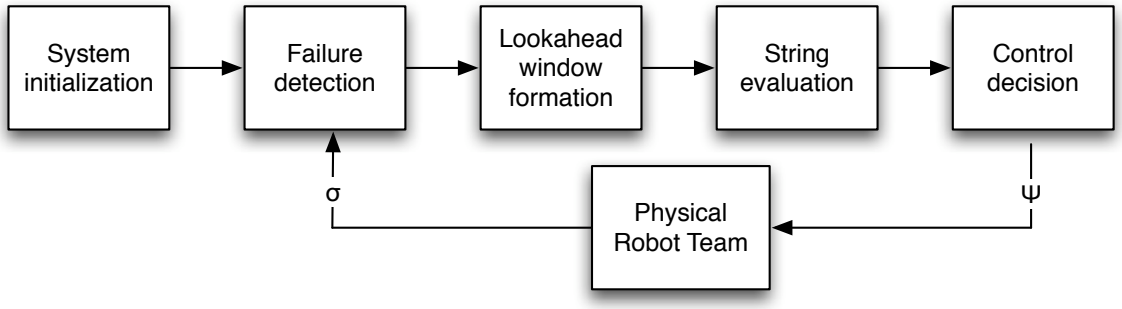


Fig. 6.10 Control algorithm with failure detection

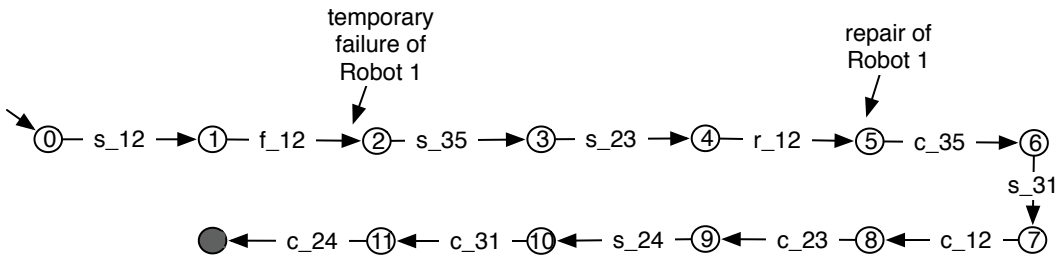


Fig. 6.11 Events executed in the UCSM after a temporary failure

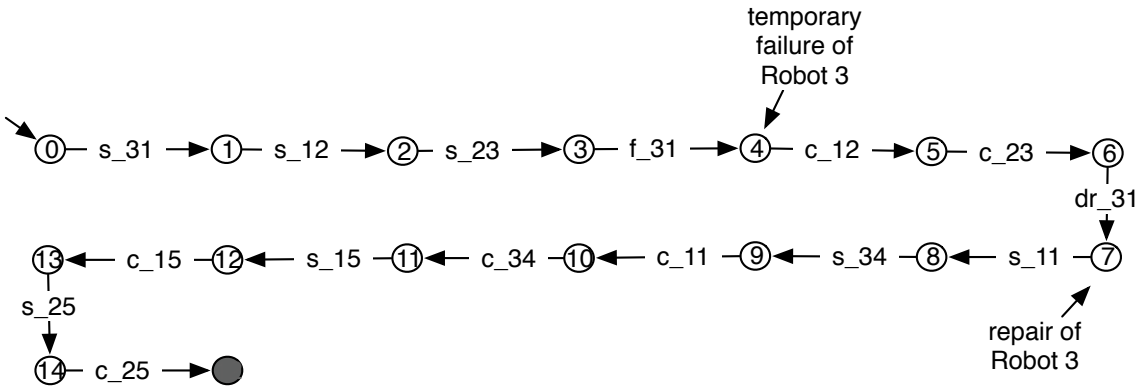


Fig. 6.12 Events executed in the UCSM after a failure with task re-initialization

6.6 Computational Results Using the Proposed Control Methodology

This section discusses the experimental results of the controller design methodology for dynamic task allocation using the warehouse patrolling application scenario. The membership functions for three selected fuzzy variables, namely endurance, efficiency and designer's choice, are shown in Table 6.3.

Using the membership functions, the ability of each robot to perform a task is computed by the fuzzy controller. The corresponding event utility function values based on Equation (6.7) are:

$$\begin{aligned} &u(start_{11}) = 0.16,\ u(start_{12}) = 0.81,\ u(start_{13}) = 0.33,\ u(start_{15}) = 0.16 \\ &u(start_{22}) = 0.33, u(start_{23}) = 0.83, u(start_{24}) = 0.81, u(start_{25}) = 0.18 \\ &u(start_{31}) = 0.81, u(start_{34}) = 0.17, u(start_{35}) = 0.83. \end{aligned} \tag{6.20}$$

The maximum profit due to control actions is equal to 4.09 and is achieved when the following task assignments are made in any order: $\{start_{31}, start_{12}, start_{23}, start_{24}, start_{35}\}$.

Two operational scenarios, one without failures and one with failures are considered for varying depth of limited lookahead windows. Three performance measures of

interest are reported. *Total utility* refers to the sum of the utility of the events executed in the experimental run. This is an indicator of the quality of task allocation decisions where the higher values indicate better task allocation decisions. *Average state space* refers to the average number of states explored each time a lookahead window is created during an experimental run. This metric refers to the computational requirement for real time decision making. Finally, *total state space* metric refers to the total number of new states explored during the entire experimental run.

Table 6.3 Fuzzy logic membership functions for the patrolling scenario

Robot	Fuzzy variable	Task 1 Member.	Task 2 Member.	Task 3 Member.	Task 4 Member.	Task 5 Member.
1	Endurance	Long	Fair	Long	-	Long
	Efficiency	Low	Medium	Medium		Medium
	Des. Choice	Low	High	Low		Low
2	Endurance	-	Fair	Fair	Fair	Short
	Efficiency		High	Medium	Medium	Medium
	Des. Choice		Low	High	High	Low
3	Endurance	Short	-	-	Fair	Fair
	Efficiency	Medium			Medium	Medium
	Des. Choice	High			Low	High

Table 6.4 shows the results of 10 experimental runs for the case without failures for limited lookahead depths (LLD) of 2, 3, 4 and 12 and Table 6.5 summarizes these results. In this scenario the sequence of events which completes the mission of patrolling the 5 warehouse regions is 12 and the size of the coupled model is 735. For the case of LLD 12, the entire states space of the coupled model is explored and the maximum possible total utility is obtained in every run. In LLDs of 3 and 6, every experimental run re-

turned the maximum total utility. While this is a strong indication that a LLD of 25% and 50% of the maximum lookahead depth can return comparable results to the maximum total utility, this performance cannot be guaranteed for every case. For the cases of LLD, the average computational requirements to generate lookahead window were 10% and 56% of the computational requirements of the maximum lookahead depth and 15% and 68% of the complete state space was explored. In the case of LLD 2, the average total utility is 75% of the maximum total utility at a small fraction of the computational requirements of the LLD 12 case. In summary, these experiments indicate that using a limited lookahead control policy for task allocation performs comparably to a control policy, which considers the total state space at a fraction of the total computational requirements.

Table 6.6 shows the results of 20 experimental run for the operational scenario with robot failures for limited LLDS of 2, 3, 6 and 12. Robot failures are generated using a uniformly distributed random variable $p_j \in [0,1]$ denoting the probability of Robot j to fail. Furthermore, a uniformly distributed random variable $R:\{failure_{jk}\} \rightarrow [1,10]$ is defined to describe the severity of a robot failure and the repair time as follows:

Failure with task re-initialization if $5 < R < 10$ (6.21)

Temporary failure if $R \leq 5$

The random variable R is also used to denote the robot repair time as follows:

$$repair\ \ time = \begin{cases} R & ,R \leq 5 \\ 10 - R & ,R > 5 \end{cases} \quad (6.22)$$

For example if $R = 4$, the failure is considered as temporary and the robot can be repaired after 4 event intervals. If R=6, the repair is considered temporary with task re-initializations and the robot can be repaired after 10-6=4 event intervals.

Table 6.7 summarizes the results of the 20 experimental run. In this scenario, the uncertain environment arising from robot failures produced a more mixed set of results. In all cases including the maximum lookahead depth of 14 (the increase from 12 is due to the robot failure and repair/task re-initialization events) the average total utility was less than the maximum total utility. As, expected the computational requirements were comparable to the scenario without failures. Note that the percentage of computational requirements for the LLD of 2, 3, and 6 were less since the maximum lookahead depth in this scenario is higher.

Table 6.4 Experimental results for the case without failures

Depth		Exp. 1	Exp. 2	Exp. 3	Exp. 4	Exp. 5	Exp. 6	Exp. 7	Exp. 8	Exp. 9	Ex. 10
2	Total Utility	4.09	4.09	3.44	3.44	2.80	3.45	3.44	2.80	2.80	3.44
	Avg. State Space	4.00	3.90	3.90	3.90	3.70	4.00	3.90	3.70	3.90	3.90
	Total state space	40	39	39	39	37	40	39	37	39	39
3	Total Utility	4.09	4.09	4.09	4.09	4.09	4.09	4.09	4.09	4.09	4.09
	Avg. State Space	14.20	13.20	15.00	14.20	14.10	14.40	13.40	14.50	13.20	15.00
	Total state space	110	103	115	110	109	111	104	112	103	115
6	Total Utility	4.09	4.09	4.09	4.09	4.09	4.09	4.09	4.09	4.09	4.09
	Avg. State Space	85.40	84.00	81.50	80.40	81.70	77.90	83.60	82.60	85.40	84.30
	Total state space	505	505	503	481	484	479	505	503	505	505
12	Total Utility	4.09	4.09	4.09	4.09	4.09	4.09	4.09	4.09	4.09	4.09
	Avg. State Space	147.40	152.40	149.90	146.70	152.20	147.40	148.50	130.10	153.20	148.10
	Total state space	735	735	735	735	735	735	735	735	735	735

Table 6.5 Summary of experimental results for the case without failures

Depth		Average	% compared with maximum lookahead depth
2	Total Utility	3.38	
	Avg. State Space	3.88	3%
	Total state space	38.8	5%
3	Total Utility	4.09	
	Avg. State Space	14.12	10%
	Total state space	109.2	15%
6	Total Utility	4.09	
	Avg. State Space	82.68	56%
	Total state space	497.5	68%
12	Total Utility	4.09	
	Avg. State Space	147.59	100%
	Total state space	735	100%

Table 6.6 Experimental results for the case with failures

Depth		Exp. 1	Exp. 2	Exp. 3	Exp. 4	Exp. 5	Exp. 6	Exp. 7	Exp. 8	Exp. 9	Ex. 10
2	Total Utility	3.44	3.44	3.44	4.09	2.8	3.44	2.8	3.44	3.45	3.45
	Avg. State Space	2.92	3.15	3.58	2.84	3.08	2.91	3.16	3	3.16	2.83
	Total state space	38	41	43	37	37	35	38	36	38	34
3	Total Utility	4.09	4.09	3.44	4.09	4.09	4.09	4.09	4.09	4.09	4.09
	Avg. State Space	14.08	15.23	14.58	13.07	13	10.92	10.16	13.69	11	12.76
	Total state space	142	165	147	142	144	120	104	148	112	142
6	Total Utility	4.09	4.09	4.09	4.09	3.44	4.09	4.09	4.09	4.09	3.44
	Avg. State Space	90.66	161.25	129.66	115.30	188.33	95.25	131.15	87.153	138	191.25
	Total state space	950	1280	1124	1171	1392	992	1218	982	1187	1372
12	Total Utility	4.09	4.09	3.6	4.09	4.09	4.09	3.44	3.6	4.09	3.6
	Avg. State Space	460.38	263.92	338.78	233.30	385.83	262.38	345.84	444.71	268.30	344.35
	Total state space	2723	2723	2723	2723	2723	2723	2723	2723	2723	2723

Table 6.6 (Continued)

Depth		Exp. 11	Exp. 12	Exp. 13	Exp. 14	Exp. 15	Exp. 16	Exp. 17	Exp. 18	Exp. 19	Exp. 20
	Total Utility	3.44	3.44	3.44	3.44	2.8	4.09	2.8	3.44	3.45	3.44
	Avg. State Space	3.08	3.61	3	3	3.16	3.38	2.92	3.15	3.16	3.46
2	Total state space	40	47	36	36	38	44	38	41	38	45
	Total Utility	4.09	3.44	4.09	4.09	4.09	4.09	4.09	4.09	4.09	4.09
	Avg. State Space	11.58	15.69	11.83	13.61	14.84	15.84	11.58	12.76	17.58	10.08
3	Total state space	119	168	121	149	163	170	119	142	176	104
	Total Utility	4.09	4.09	3.44	4.09	4.09	3.45	4.09	4.09	4.09	3.45
	Avg. State Space	134.91	149.38	175.33	132.75	132.23	93.33	96.75	180.38	128.30	96
6	Total state space	1158	1285	1332	1155	1220	978	1007	1374	1194	998
	Total Utility	3.60	4.09	4.09	4.09	3.44	3.45	4.09	4.09	3.59	4.09
	Avg. State Space	229.33	443.66	446.08	272.66	475.58	245.15	653.5	539	252.58	232
12	Total state space	2723	2723	2723	2723	2723	2723	2723	2723	2723	2723

Table 6.7 Summary of experimental results for the case with failures

Depth		Average	% compared with maximum lookahead depth
2	Total Utility	3.38	
	Avg. State Space	3.13	1%
	Total state space	39.00	1%
3	Total Utility	4.03	
	Avg. State Space	13.20	4%
	Total state space	139.85	5%
6	Total Utility	3.93	
	Avg. State Space	132.37	37%
	Total state space	1168.45	43%
14	Total Utility	3.87	
	Avg. State Space	356.87	100%
	Total state space	2723.00	100%

To demonstrate the impact of the LLD on task allocation decisions based on the total utility of the executed events, we compared the null hypothesis $H_o : \mu_2 = \mu_3 = \mu_6 = \mu_{14}$ against the alternative hypothesis that the means are different using a single-factor ANOVA with four levels of LLD and 20 replications. The ANOVA is summarized in

Table 6.8. The results indicate that there is a significant difference between the means and LLD is a factor that impacts the total utility level.

Subsequently we conducted similar experiment but this time using 3 levels of LLD 3, 6 and 14. The ANOVA for this experimental deign is summarized in Table 6.9. The results of this analysis show no significant difference between the means.

Table 6.8 ANOVA for total utility with 4 levels of LLD

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*
Between Groups	4.99	3	1.66	20.30	9.18E-10
Within Groups	6.23	76	0.08		
Total	11.22	79			

*$F_{0.05,3,76} = 2.725$

Table 6.9 ANOVA for total utility with 3 levels of LLD

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*
Between Groups	0.24	2	0.12	1.82	0.17
Within Groups	3.82	57	0.06		
Total	4.06	59			

*$F_{0.05,2,57} = 3.159$

In summary, these preliminary experiments indicate that a limited lookahead control policy provides a computational efficient approach to the problem of task allocation. However, the depth of the limited lookahead window must be carefully chosen based on the characteristics of the mission as well as the desired level of optimality as the quality of the task allocations is dependent on this parameter, Furthermore, these preliminary results indicate that LLD of less than ¼ of the maximum lookahead depth provides compounded reductions in the computational requirements where on average less than 10% of the computational requirements are sufficient to make control decisions. However, it

must be pointed out that this statement must be further verified with larger experimental designs.

6.7 Conclusions

In this chapter we describe a novel control methodology for task allocation in cooperative robot teams. Finite automata formalism is used to model the robot team and the mission requirements as discrete event systems. In developing the system model, we considered flexibility in task assignment, robot coordination for task completion and robot failures and repairs. These characteristics are commonly encountered in mission planning and execution of cooperative robot teams. We also describe a utility function for task allocation that uses fuzzy logic to describe various robot capabilities which are difficult to quantify, Subsequently, a limited lookahead control policy coupled with a fuzzy controller is developed for task allocation in real time.

The use of limited lookahead policies presents significant advantages in terms of computational complexity. The computationally complexity of the traditional supervisory control problem is polynomial [107] if the UCSM and the specification's model describe *perfectly* the behavior of the robot team and the mission requirements. However, if there is *imperfect* information about the system then the complexity becomes PSPACE hard [134]. The limited lookahead policy proposed in this chapter is based on the coupled model of the system where the computational complexity of the coupled model generation is linear. The computational results show that only a fraction of the coupled model

needs to be explored to in the limited lookahead window to make task allocation decisions. The preliminary results show that task allocation decisions based on the limited lookahead window control policy produces comparable results when compared with exploring the complete coupled model. Further completely randomized experimentation is required to generalize these findings.

The limited lookahead depth is a critical parameter affecting the quality of task allocation as well as computational complexity. The results in Section 6.6 indicate that larger limited lookahead depths lead to higher number of states visited by the control algorithm. This is an expected result, however, the associated increase in the utility of task allocation is not as clear cut. In this work, as in most of the referenced literature, the depth of the lookahead window is arbitrarily chosen. In [115], the depth window is computed based on the number of uncontrollable events in the system. A future research direction involves determining the characteristic associated with cooperative robot teams and their missions, which may be used to develop a methodology to calculate a dynamic limited lookahead depth in real time. Such an approach will result in a controller that is adaptable to the changing needs of missions and cooperative robot teams.

In the described patrolling application, robots carry multiple sensors to patrol one or more regions of a warehouse. Consider Robot 3 that carries a fire detection sensor and a vision system. A failure of the fire detection sensor can be considered by the control methodology as a partial failure of the robot and a task that does not require fire detection capabilities such as Task 4 can still be assigned to that robot. In this manner the robot is considered partially functional and its ability to perform a task is recomputed by the

fuzzy controller presented in Section 4, leading to intelligent utilization of system resources.

In addition to partial failures, this work may be extended to include catastrophic failures where the robot will not re-join the team. Pertaining to the warehouse patrolling scenario, consider the event sequence depicted in Fig. 6.13. Robot 1 is assigned to Task 5, which completes in State 2. Task 5 can be assigned to Robot 3 or to Robot 1 and Robot 2 in the particular sequence. In State 7, a catastrophic failure event is initiated for Robot 2. Since Robot 2 will never be repaired, Task 5 will never be completed unless Task 5 is reset. Thus, any string initiated in State 8 of the figure will not reach a marked state and the team's mission will not be completed. State 8 is called blocking state. Future research involves developing a methodology that will address issues related to state blocking.

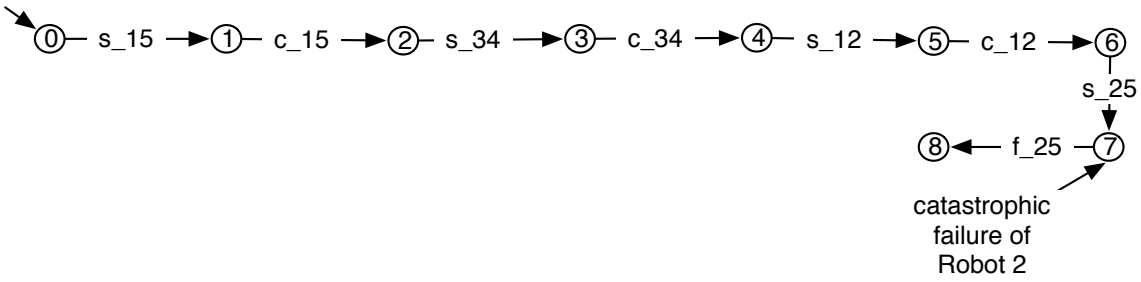


Fig. 6.13 Blocking example

Chapter 7

Conclusions and Future Work

Several industrial, military and civilian applications require autonomous multi robot solutions. Along with each robot's capabilities to perform basic functions such as navigation and obstacle avoidance, a key characteristic of multi robot systems is that of cooperation and coordination. The goal of this research is to design functional components common to a wide range of autonomous mobile robot teams and investigate their deployment under the umbrella of a hybrid control architecture. This research concentrates on the development of four main functional components: Navigation and Obstacle Avoidance, Task Achieving Behaviors, Localization and Mission Planning. Chapter 3 described a method for navigation and obstacle avoidance using range sensors and a stereo vision system. Chapter 4 presented a target tracking method, while Chapter 5 described a localization method based on Fuzzy Logic and Kalman Filtering. Finally, Chapter 6 presented a mission planning method for robot coordination and accommodation of resource failures.

At the beginning of this research, there were questions in our minds regarding the ability of a robot team to efficiently and effectively perform complex missions such as warehouse patrolling. When the development of a functional component was completed,

it seemed that we were coming closer to the final goal, but at the same time, new challenges such as sensor operation in various environments and new ideas such as partial robot failures were uncovered. We were able to address a number of new challenges. The rest, gave us directions for future research. Even though a real patrolling scenario has not been implemented yet, the advances of our work suggest that this is possible.

Our vision is that autonomous robot teams are able to complete missions that involve cooperation and coordination, efficiently and effectively. This work demonstrates results that support and broaden the vision to include intelligent robots that can adapt to mission requirements in dynamic and uncertain environments. In addition, the diversity between the team members can extend the application domains from the industrial/civilian environments to military applications in adversarial environments.

The rest of this chapter summarizes the specific contributions of this research to date and the future work.

7.1 Contributions

7.1.1 Navigation and Obstacle Avoidance

Key characteristic of autonomous patrolling applications is that of navigation and obstacle avoidance. Each robot of the team should demonstrate the capability of navigating in a facility and avoid collision with surrounding objects. In Chapter 3, a navigation and obstacle avoidance method for mobile robots based on ultrasonic and vision sensors

has been presented. Main contribution of this work is the utilization of the stereo vision system as range sensor. A depth estimation method with uncalibrated cameras was introduced that uses the image size, the angle of view of the cameras and the relative position of two different captures of the same scene. Then, range information from both the ultrasonic sensors and the vision system have been used in parallel to safely navigate a mobile robot. Parallel sensor utilization is particularly useful in patrolling applications where lighting variations in different areas may affect the operation of the vision system but not the operation of the ultrasonic sensors. The Navigation and Obstacle avoidance approach outlined in Chapter 3 appears in [100] and [135].

7.1.2 Task Achieving Behaviors: Target Tracking

Patrolling applications require the ability to recognize and track unauthorized presence in a secure facility. As Task Achieving Behavior, a target tracking method for mobile robots was developed in Chapter 4 and presented in [136] and [137]. The main differences and advantages of the presented approach compared to related research are:

1. Both cameras of the vision system track a target independently, providing a redundant mechanism that helps avoiding loosing the target. This means that even if one camera ‘loses the target’, it can retrieve information from the other camera to find it again.

2. The robot's direction is controlled by the pan/tilt angles of the cameras, allowing the robot to avoid obstacles and keep tracking a target, or to keep tracking a target that moves on uneven terrains.

7.1.3 Localization

Localization is an essential functionality for mobile robot applications such as autonomous patrolling. The processes of coordination and task allocation require exact knowledge of the position of all the robots in the team. Chapter 5 presented a localization method based on Fuzzy Logic and Kalman Filtering. The major contribution of this work is the number of sensors used. While most work in literature utilizes up to three different sensors for robot localization, work in Chapter 5 utilizes five different sensors: GPS, IMU, odometer, stereo vision system and laser range finder. In addition, the Fuzzy Logic controller design allows for incorporation of the error in sensor readings into the EKF without the use of an error model. The error in sensor readings is approximated by multiple zero mean Gaussian distributions and it is included in the covariance matrix of the measurement model. The localization method described in Chapter 5 appears in [138].

7.1.4 Mission Planning

Chapter 6 is dedicated to Mission Planning. A general task allocation supervisory controller has been developed to oversee the team of mobile robots that work together to

perform applications such as warehouse patrolling. The control methodology presented is partially based on the RW supervisory control theory. However, the controller utilizes a limited lookahead policy to enable and disable events in the system based on the evaluation of a fuzzy logic based utility function and robot availability.

Contributions of this work focus in the areas of computational complexity and of task allocation. In traditional supervisory control theory the acceptable event sequences are computed apriori. However, as the problem size increases so does the state space of the supervisor. In addition, applications with significant degree of uncertainty are highly dynamic and require frequent modifications to the control model. In this work, instead of constructing the complete supervisor, a limited lookahead policy is utilized that constructs a lookahead supervisor based on the evolution of the system in real time. The proposed methodology offers an advantage in highly dynamic systems, since the control model can change on the fly.

Concerning the task allocation problem, most work in literature utilizes random or biding methodologies to assign robots to tasks. This work proposes a novel approach that uses a fuzzy logic controller, which introduces a measure for the ability of a robot to perform a task based on criteria such as endurance, efficiency and designer's choice. The supervisory control design presented in Chapter 6 appears in [139] while an earlier version of the controller was published in [140].

7.2 Future Work

Additional features can be incorporated to the design of the functional components to improve the performance of the patrolling system. Future work involves:

1. Enhancement of the Localization component to identify random objects in the environment the robot travels to use as landmarks for the range sensors. However, including multiple landmarks into the system's model increases the state space of the localization problem. For this reason a methodology such as compressed Kalman filtering is required to decrease the state space due to incorporation of multiple landmarks in the system model. The compressed Kalman filtering methodology will remove states of the system's model that are no longer active.
2. Mission Planning enhancements involve the development of a controller tolerant to partial and catastrophic failures of the robot team members. In the presented configuration, when a failure in a robot's sensor occurs, the robot is considered failed by the controller. However, the failed robot may still be useful in a different task that does not require the failed sensor. In addition to partial failures, there are situations where during the mission a robot may fail and will never re-join the team due to a catastrophic failure. Catastrophic failures result in blocking states, from where the system will not be able to move on. The Mission Planning component should be able to avoid blocking states.
3. Deployment of the robot team in an actual patrolling application. So far, the performance of most of the functional components has been verified and validated

experimentally. However, the performance of the Mission Planning component has been validated in a simulated setting.

References

[1] G. A. Bekey, *Autonomous Robots: From biological inspiration to implementation and control*, MIT Press, 2005.

[2] C. Cheng, W. Han, C. Ng Teck, J. Ibanez-Guzman, J. Shen, and W. Chan Chun, "Target-tracking and path planning for vehicle following in jungle environment," in *8th Control, Automation, Robotics and Vision Conference*, 2004, pp. 455-460, vol.4.

[3] Y. Zhang, M. Schervish, E. U. Acar, and H. Choset, "Probabilistic methods for robotic landmine search," in *Proceedings of International Conference on Intelligent Robots and Systems*, 2001, pp. 1525-1532 vol.3.

[4] V. Kumar and F. Sahin, "Cognitive maps in swarm robots for the mine detection application," in *IEEE International Conference on Systems, Man and Cybernetics* 2003, pp. 3364-3369, vol.4.

[5] J. S. Jennings, G. Whelan, and W. F. Evans, "Cooperative search and rescue with a team of mobile robots," in *8th International Conference on Advanced Robotics*, 1997, pp. 193-200.

[6] N. Ruangpayoongsak, H. Roth, and J. Chudoba, "Mobile robots for search and rescue," in *IEEE International Workshop on Safety, Security and Rescue Robotics*, 2005, pp. 212-217.

[7] K. Kaaniche, B. Champion, C. Pegard, and P. Vasseur, "A vision algorithm for dynamic detection of moving vehicles with a UAV," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, 2005, pp. 1878-1883.

[8] M. Veloso and P. Stone, " Individual and collaborative behaviors in a team of homogeneous robotic soccer agents " in *roceedings of the Third International Conference on Multi-Agent Systems*, 1998, pp. 309-316.

[9] L. E. Parker, "ALLIANCE: an architecture for fault tolerant multirobot cooperation," *IEEE Transactions on Robotics and Automation,* 1998, pp. 220-240, vol.14.

[10] M. Long, A. Gage, R. Murphy, and K. Valavanis, "Application of the distributed field robot architecture to a simulated demining task," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, 2005, pp. 3193-3200.

[11] J. S. Albus, H. G. McCain, and R. Lumia, "NASA/NBS standard reference model for telerobot control system architecture (NASREM)," in *Technical Report, Robots Systems Division, National Bureau of Standards*, 1987.

[12] J. E. Naranjo, C. Gonzalez, T. de Pedro, R. Garcia, J. Alonso, and M. A. Sotelo, "AUTOPIA architecture for automatic driving and maneuvering," in *IEEE Intelligent Transportation Systems Conference*, 2006, pp. 1220-1225.

[13] H. A. Hagras, "A hierarchical type-2 fuzzy logic control architecture for autonomous mobile robots," *IEEE Transactions on Fuzzy Systems,* 2004, pp. 524-539, vol.12.

[14] H. Hu, J. M. Brady, J. Grothusen, F. Li, and P. J. Probert, "LICAs: a modular architecture for intelligent control of mobile robots," in *International Conference on Intelligent Robots and Systems 95. 'Human Robot Interaction and Cooperative Robots'*, 1995, pp. 471-476 vol.1.

[15] R. Stenzel, "A behavior-based control architecture," in *IEEE International Conference on Systems, Man, and Cybernetics*, 2000, pp. 3235-3240 vol.5.

[16] R. Brooks, "A robust layered control system for a mobile robot," *IEEE Journal of Robotics and Automation,* 1986, pp. 14-23, vol.2.

[17] A. H. P. Selvatici and A. H. R. Costa, "A hybrid adaptive architecture for mobile robots based on reactive behaviors," in *Fifth International Conference on Hybrid Intelligent Systems*, 2005.

[18] B. Lenser and M. Veloso, "A modular hierarchical behavior based architecture," in *RoboCup 2001: Robot Soccer World Cup V*, 2002.

[19] Y. Yan, Q. Zhu, and C. Cai, "Hybrid control architecture of mobile robot based on subsumption architecture," in *Proceedings of the 2006 IEEE International Conference on Mechatronics and Automation*, 2006, pp. 2168-2172.

[20] M. Lindstrom, A. Oreback, and H. I. Christensen, "BERRA: a research architecture for service robots," in *IEEE International Conference on Robotics and Automation*, 2000, pp. 3278-3283, vol.4.

[21] J. Wang, "DRS operating primitives based on distributed mutual exclusion," in *International Conference on Intelligent Robots and Systems '93*, 1993, pp. 1085-1090, vol.2.

[22] H. Asama, A. Matsumoto, and Y. Ishida, "Design of an autonomous and distributed robot system: Actress," in *International Workshop on Intelligent Robots and Systems '89. The Autonomous Mobile Robots and Its Applications*, 1989, pp. 283-290.

[23] T. Huntsberger, P. Pirjanian, A. Trebi-Ollennu, H. Das Nayar, H. Aghazarian, A. J. Ganino, M. Garrett, S. S. Joshi, and P. S. Schenker, "CAMPOUT: a control architecture for tightly coupled coordination of multirobot systems for planetary surface exploration," *IEEE Transactions on Systems, Man and Cybernetics, Part A,,* 2003, pp. 550-559, vol.33.

[24] P. Caloud, C. Wonyun, J. C. Latombe, C. Le Pape, and M. Yim, "Indoor automation with many mobile robots," in *Intelligent Robots and Systems '90. 'Towards a New Frontier of Applications'*, 1990, pp. 67-72, vol.1.

[25] E. Gat, "On Three-Layer Architectures," in *Kortenkamp, D., Banasso, R.P. Artificial Intelligence and Mobile Robots*: MIT Press, 1997, pp. 195-210.

[26] H. Secchi, V. Mut, R. Carelli, M. Schneebeli, and T. Bastos, "A hybrid Control Architecture for Mobile Robots. Classic Control, Behavior Based Control and Petri Nets."

[27] A. V. Timofeev, F. A. Kolushev, and A. A. Bogdanov, "Hybrid algorithms of multi-agent control of mobile robots," in *International Joint Conference on Neural Networks*, 1999, pp. 4115-4118, vol.6.

[28] L. Doitsidis, K. P. Valavanis, and N. C. Tsourveloudis, "Fuzzy logic based autonomous skid steering vehicle navigation," in *IEEE International Conference on Robotics and Automation*, 2002, pp. 2171-2177, vol.2.

[29] A. N. Rajagopalan, S. Chaudhuri, and M. Uma, "Depth estimation and image restoration using defocused stereo pairs," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 2004, pp. 1521-1525, vol.26.

[30] U. Mudenagudi and S. Ghaudhuri, "Depth estimation using defocused stereo image pairs," in *The Proceedings of the Seventh IEEE International Conference on Computer Vision*, 1999, pp. 483-488, vol.1.

[31] K. Umeda and T. Takahashi, "Subpixel stereo method: a new methodology of stereo vision," in *IEEE International Conference on Robotics and Automation*, 2000, pp. 3215-3220, vol.4.

[32] J. Fan and T. E. Weymouth, "Depth from dynamic stereo images," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1989, pp. 250-255.

[33] L. Shang-Hong, F. Chang-Wu, and C. Shyang, "A generalized depth estimation algorithm with a single image," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 1992, pp. 405-411, vol.14.

[34] E. Malis and P. Rives, "Robustness of image-based visual servoing with respect to depth distribution errors," in *IEEE International Conference on Robotics and Automation*, 2003, pp. 1056-1061, vol.1.

[35] S. Derrouich, K. Izumida, and K. Shiiya, "A combination of monocular CCD camera and inertial-sensor for range estimation," in *IEEE 2002 28th Annual Conference of the Industrial Electronics Society*, 2002, pp. 2191-2196, vol.3.

[36] M. Sonka, V. Hlavac, and R. Boyle, *Image processing, analysis and machine vision*: PWS Publishing, 1999.

[37] E. Sahin and P. Gaudiano, "Mobile robot range sensing through visual looming," in *Proceedings of Intelligent Control,* 1998, pp. 370-375.

[38] I. Ohya, A. Kosaka, and A. Kak, "Vision-based navigation by a mobile robot with obstacle avoidance using single-camera vision and ultrasonic sensing," *IEEE Transactions on Robotics and Automation,* 1998, pp. 969-978, vol.14.

[39] B. R. Duffy, C. Garcia, C. F. B. Rooney, and G. M. P. O'Hare, "Sensor fusion for social robotics," in *31st Int. Symp. On Robotics*, Montréal, Canada, 2000, pp. 258-264.

[40] R. C. Mann, J. P. Jones, M. Beckerman, C. W. Glover, L. Farkas, J. Han, E. Wacholder, and J. R. Einstein, "An intelligent integrated sensor system for the ORNL mobile robot," in *IEEE International Symposium on Intelligent Control*, 1988, pp. 170-173.

[41] G. Gubber and H. Sahli, "Sensor integration on a mobile robot," in *12th International Symposium on Measurement and Control in Robotics*, France, 2002.

[42] C. Jennings, D. Murray, and J. J. Little, "Cooperative robot localization with vision-based mapping," in *IEEE International Conference on Robotics and Automation*, 1999, pp. 2659-2665, vol.4.

[43] V. Tucakov, M. Sahota, D. Murray, A. Mackworth, J. Little, S. Kingdon, C. Jennings, and R. Barman, "Spinoza: a stereoscopic visually guided mobile robot," in *Proceedings of the Thirtieth Hawaii International Conference on System Sciences*, 1997, pp. 188-197, vol.5.

[44] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine,* 1997, pp. 23-33, vol.4.

[45] W. D. Rencken, "Autonomous sonar navigation in indoor, unknown and unstructured environments," in *International Conference on Intelligent Robots and Systems '94*, 1994, pp. 431-438, vol.1.

[46] A. Elfes, "Sonar-based real-world mapping and navigation," *IEEE Journal of Robotics and Automation,* 1987, pp. 249-265, vol.3.

[47] K. P. Valavanis, T. Hebert, R. Kolluru, and N. Tsourveloudis, "Mobile robot navigation in 2-D dynamic environments using an electrostatic potential field," *IEEE Transactions on Systems, Man and Cybernetics, Part A,* 2000, pp. 187-196, vol.30.

[48] CCIR, "Encoding parameter of digital television for studios," International Radio Consultative Committee 1990.

[49] J. J. De Dios and N. Garcia, "Face detection based on a new color space YCgCr," in *International Conference on Image Processing*, 2003, pp. III-909-12 vol.2.

[50] R. Jain, R. Kasturi, and B. G. Schunck, *Machine vision*: McGraw-Hill International Editions, 1995.

[51] J. C. Russ, *The image processing handbook*: IEEE PRESS, 1995.

[52] J. Ding, H. Kondou, H. Kimura, Y. Hada, and K. Takase, "Robust tracking for camera control on an irregular terrain vehicle," in *Proceedings of the 41st SICE Annual Conference*, 2002, pp. 1191-1196, vol.2.

[53] P. Saeedi, P. Lawrence, and D. Lowe, "3D motion tracking of a mobile robot in a natural environment," in *IEEE International Conference on Robotics and Automation*, 2000, pp. 1682-1687, vol.2.

[54] C. Balkenius and L. Kopp, "Visual tracking and target selection for mobile robots," in *Proceedings of the First Euromicro Workshop on Advanced Mobile Robot*, 1996, pp. 166-171.

[55] H. Y. Wang, S. Itani, T. Fukao, and N. Adachi, "Image-based visual adaptive tracking control of nonholonomic mobile robots," in *International Conference on Intelligent Robots and Systems*, 2001, pp. 1-6, vol.1.

[56] L. Fu-Chang, T. Wei, and S. L. Tzuu-Hseng, "An experimental study on tracking control of two autonomous mobile robots," in *23rd International Conference on Industrial Electronics, Control and Instrumentation*, 1997, pp. 1311-1316, vol.3.

[57] H. M. Gross, H. J. Boehme, and T. Wilhelm, "Contribution to vision-based localization, tracking and navigation methods for an interactive mobile service-robot," in *IEEE International Conference on Systems, Man, and Cybernetics*, 2001, pp. 672-677, vol.2.

[58] M. Kleinehagenbrock, S. Lang, J. Fritsch, F. Lomker, G. A. Fink, and G. Sagerer, "Person tracking with a mobile robot based on multi-modal anchoring," in *11th IEEE International Workshop on Robot and Human Interactive Communication*, 2002, pp. 423-429.

[59] T. Darrell, G. Gordon, M. Harville, and J. Woodfill, "Integrated person tracking using stereo, color, and pattern detection," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1998, pp. 601-608.

[60] L. Yong-Beom, Y. Bum-Jae, and L. Seong-Whan, "A real-time color-based object tracking robust to irregular illumination variations," in *IEEE International Conference on Robotics and Automation*, 2001, pp. 1659-1664, vol.2.

[61] E. B. Meier and F. Ade, "Using the condensation algorithm to implement tracking for mobile robots," in *Third European Workshop on Advanced Mobile Robots*, 1999, pp. 73-80.

[62] J. Miyata, T. Murakami, and K. Ohnishi, "An approach to tracking motion of mobile robot for moving object," in *26th Annual Confference of the IEEE Industrial Electronics Society*, 2000, pp. 2249-2254, vol.4.

[63] L. Sung-On, C. Young-Jo, H.-B. Myung, Y. Bum-Jae, and O. Sang-Rok, "A stable target-tracking control for unicycle mobile robots," in *International Conference on Intelligent Robots and Systems*, 2000, pp. 1822-1827, vol.3.

[64] T. Wei and S. L. Tzuu-Hseng, "Realization of two-dimensional target tracking problem via autonomous mobile robots using fuzzy sliding mode control," in *Proceedings of the 24th Annual Conference of the IEEE Industrial Electronics Society*, 1998, pp. 1158-1163, vol.2.

[65] Z. Lin, V. Zeman, and R. V. Patel, "On-line robot trajectory planning for catching a moving object," in *IEEE International Conference on Robotics and Automation*, 1989, pp. 1726-1731, vol.3.

[66] H. Rowley, S. Baluja, and T. Kanade, "Neural network based face detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, 1997, pp. 690-696.

[67] T. H. S. Li, C. Shih-Jie, and T. Wei, "Fuzzy target tracking control of autonomous mobile robots by using infrared sensors," *IEEE Transactions on Fuzzy Systems,* vol. 12, pp. 491-501, 2004.

[68] R. C. Luo and C. Tse Min, "Autonomous mobile target tracking system based on grey-fuzzy control algorithm," *IEEE Transactions on Industrial Electronics,* vol. 47, pp. 920-931, 2000.

[69] F. J. Montecillo-Puente, V. Ayala-Ramirez, A. Perez-Garcia, and R. E. Sanchez-Yanez, "Fuzzy color tracking for robotic tasks," in *IEEE International Conference on Systems, Man and Cybernetics*, 2003, pp. 2769-2773, vol.3.

[70] S. Buluswar and B. Draper, "Color Machine Vision for Autonomous Vehicles," *Engineering Applications of Artificial Intelligence,* vol. 11, pp. 245-256, 1998.

[71] M. J. Swain and D. H. Ballard, "Indexing via color histograms," in *Third International Conference on Computer Vision*, 1990, pp. 390-393.

[72] C. Hsin-Chia, C. Wei-Jung, and W. Sheng-Jyh, "Contrast based color segmentation with adaptive thresholds," in *International Conference on Image Processing*, 2002, pp. II-73-II-76, vol.2.

[73] C. Yong and C. Zhu, "Obstacle detection using adaptive color segmentation and planar projection stereopsis for mobile robots," in *IEEE International Conference on Robotics, Intelligent Systems and Signal Processing*, 2003, pp. 1097-1101, vol.2.

[74] B. V. Funt and G. D. Finlayson, "Color constant color indexing," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 1995, pp. 522-529, vol.17.

[75] W. Ying and T. S. Huang, "Nonstationary color tracking for vision-based human-computer interaction," *IEEE Transactions on Neural Networks,* 2002, pp. 948-960, vol.13.

[76] J. Orwell, P. Remagnino, and G. A. Jones, "Multi-camera colour tracking," in *Second IEEE Workshop on Visual Surveillance*, 1999, pp. 14-21.

[77] K. Sobottka and I. Pitas, "Segmentation and tracking of faces in color images," in *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition*, 1996, pp. 236-241.

[78] A. Arsenio and M. I. Ribeiro, "Active range sensing for mobile robot localization," in *International Conference on Intelligent Robots and Systems*, 1998, pp. 1066-1071, vol.2.

[79] H. Huosheng and G. Dongbing, "Landmark-based navigation of mobile robots in manufacturing," in *7th IEEE International Conference on Emerging Technologies and Factory Automation*, 1999, pp. 121-128, vol.1.

[80] H. Chou, M. Traonmilin, E. Ollivier, and M. Parent, "A simultaneous localization and mapping algorithm based on Kalman filtering," in *IEEE Intelligent Vehicles Symposium*, 2004, pp. 631-635.

[81] U. Larsson, J. Forsberg, and A. Wernersson, "Mobile robot localization: integrating measurements from a time-of-flight laser," *IEEE Transactions on Industrial Electronics,* 1996, pp. 422-431, vol.43.

[82] T. Ching-Chih, "A localization system of a mobile robot by fusing dead-reckoning and ultrasonic measurements," in *IEEE Instrumentation and Measurement Technology Conference*, 1998, pp. 144-149, vol.1.

[83] P. Goel, S. I. Roumeliotis, and G. S. Sukhatme, "Robust localization using relative and absolute position estimates," in *International Conference on Intelligent Robots and Systems*, 1999, pp. 1134-1140, vol.2.

[84] B. Barshan and H. F. Durrant-Whyte, "Inertial navigation systems for mobile robots," *IEEE Transactions on Robotics and Automation,* 1995, pp. 328-342, vol.11.

[85] S. Niwa, T. Masuda, and Y. Sezaki, "Kalman filter with time-variable gain for a multisensor fusion system," in *International Conference on Multisensor Fusion and Integration for Intelligent Systems*, 1999, pp. 56-61.

[86] K. Seong-Baek, L. Seung-Yong, H. Tae-Hyun, and C. Kyoung-Ho, "An advanced approach for navigation and image sensor integration for land vehicle navigation," in *IEEE 60th Vehicular Technology Conference*, 2004, pp. 4075-4078, vol. 6.

[87] E. T. Baumgartner and S. B. Skaar, "An autonomous vision-based mobile robot," *IEEE Transactions on Automatic Control,* 1994, pp. 493-502, vol.39.

[88] M. Marron, J. C. Garcia, M. A. Sotelo, E. Lopez, and M. Mazo, "Fusing odometric and vision data with an EKF to estimate the absolute position of an autonomous mobile robot," in *IEEE Conference Emerging Technologies and Factory Automation*, 2003, pp. 591-596, vol.1.

[89] F. Chenavier and J. L. Crowley, "Position estimation for a mobile robot using vision and odometry," in *IEEE International Conference on Robotics and Automation*, 1992, pp. 2588-2593, vol.3.

[90] E. Stella, G. Cicirelli, F. P. Lovergine, and A. Distante, "Position estimation for a mobile robot using data fusion," in *IEEE International Symposium on Intelligent Control*, 1995, pp. 565-570.

[91] S. Kai-Tai and T. Wen-Hui, "Environment perception for a mobile robot using double ultrasonic sensors and a CCD camera," *IEEE Transactions on Industrial Electronics,* 1996, pp. 372-379, vol.43.

[92] W. S. Wijesoma, K. R. S. Kodagoda, and A. P. Balasuriya, "A laser and a camera for mobile robot navigation," in *7th International Conference on Control, Automation, Robotics and Vision*, 2002, pp. 740-745, vol.2.

[93] K. O. Arras, N. Tomatis, and R. Siegwart, "Multisensor on-the-fly localization using laser and vision," in *International Conference on Intelligent Robots and Systems*, 2000, pp. 462-467, vol.1.

[94] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of robot motion, theory, algorithms and implementation*: The MIT Press, 2005.

[95] J. Z. Sasiadek and P. Hartana, "Sensor data fusion using Kalman filter," in *Third International Conference on Information Fusion*, 2000, pp. WED5/19-WED5/25, vol.2.

[96] P. J. Escamilla-Ambrosio and N. Mort, "A hybrid Kalman filter-fuzzy logic architecture for multisensor data fusion," in *IEEE International Symposium on Intelligent Control*, 2001, pp. 364-369.

[97] R. Carrasco and A. Cipriano, "Fuzzy logic based nonlinear Kalman filter applied to mobile robots modelling," in *IEEE International Conference on Fuzzy Systems*, 2004, pp. 1485-1490, vol.3.

[98] F. Martia, A. Jimenez, D. Rodriguez-Losada, and B. M. Al-Hadithi, "A novel fuzzy Kalman filter for mobile robots localization," in *Information Processing and Management of Uncertainty in Knowledge-Based Systems*, 2004.

[99] A. Tiano, A. Zirilli, and F. Pizzocchero, "Application of interval and fuzzy techniques to integrated navigation systems," in *Joint 9th IFSA World Congress and 20th NAFIPS International Conference*, 2001, pp. 13-18, vol.1.

[100] A. Tsalatsanis, K. Valavanis, and N. Tsourveloudis, "Mobile robot navigation using sonar and range measurements from uncalibrated cameras," in *14th Mediterranean Conference on Control and Automation*, 2006, pp. 1-7.

[101] J. P. Snyder, *A working manual*. Washington: United States Government Printing Office, 1987.

[102] V. Kumar and F. Sahin, "Cognitive maps in swarm robots for the mine detection application," in *IEEE International Conference on Systems, Man and Cybernetics.*, 2003, pp. 3364-3369, vol.4.

[103] Y. Zhang, M. Schervish, E. U. Acar, and H. A. C. H. Choset, "Probabilistic methods for robotic landmine search," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems.*, 2001, pp. 1525-1532, vol.3.

[104] J. S. Jennings, G. Whelan, and W. F. Evans, "Cooperative search and rescue with a team of mobile robots," in *Proceedings of 8th International Conference on Advanced Robotics, ICAR '97,* 1997, pp. 193-200.

[105] B. Krishnamurthy and J. Evans, "HelpMate: A robotic courier for hospital use," in *IEEE International Conference on Systems, Man and Cybernetics*, 1992, pp. 1630-1634, vol.2.

[106] L. E. Parker, "ALLIANCE: an architecture for fault tolerant multirobot cooperation," *Robotics and Automation, IEEE Transactions on,* 1998, pp. 220-240, vol.14.

[107] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM Journal on Control and Optimization,* 1987, pp. 452-462, vol.18.

[108] Y. Chevaleyre, "Theoretical analysis of the multi-agent patrolling problem," in *IEEE/WIC/ACM International Conference on Intelligent Agent Technology. IAT*, 2004, pp. 302-308.

[109] R. Zlot, A. Stentz, M. B. Dias, and S. A. T. S. Thayer, "Multi-robot exploration controlled by a market economy," in *Proceedings of IEEE International Conference on Robotics and Automation. ICRA.*, 2002, pp. 3016-3023.

[110] A. V. Timofeev, F. A. Kolushev, and A. A. Bogdanov, "Hybrid algorithms of multi-agent control of mobile robots," in *International Joint Conference on Neural Networks. IJCNN*, 1999, pp. 4115-4118, vol.6.

[111] S. C. Botelho and R. Alami, "M+: a scheme for multi-robot cooperation through negotiated task allocation and achievement," in *Proceedings of IEEE International Conference on Robotics and Automation.*, 1999, pp. 1234-1239, vol.2.

[112] B. P. Gerkey and M. J. Mataric, "Sold!: auction methods for multirobot coordination," *IEEE Transactions on Robotics and Automation,* 2002, pp. 758-768, vol.18.

[113] M. G. Lagoudakis, M. Berhault, S. Koenig, P. A. K. P. Keskinocak, and A. J. A. K. A. J. Kleywegt, "Simple auctions with performance guarantees for multi-robot task allocation," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. (IROS 2004).* 2004, pp. 698-705, vol.1.

[114] R. Davis and R. G. Smith, "Negotiation as a metaphor for distributed problem solving," *Artificial Intelligence,* 1983, pp. 63-109, vol.20.

[115] S. L. Chung, S. Lafortune, and F. Lin, "Limited lookahead policies in supervisory control of discrete event systems," *IEEE Transactions on Automatic Control,* 1992, pp. 1921-1935, vol.37.

[116] R. Kumar, H. M. Cheung, and S. I. Marcus, "Extension based limited lookahead supervision of discrete event systems," *Automatica,* 1998, pp. 1327-1344, vol.34.

[117] S.-L. Chung, S. Lafortune, and F. Lin, "Recursive computation of limited lookahead supervisory controls for discrete event systems," *Discrete Event Dynamic Systems,* 1993, pp. 71-100, vol.3.

[118] S.-L. Chung, S. Lafortune, and F. Lin, "Supervisory control using variable lookahead policies," *Discrete Event Dynamic Systems,* 1994, pp. 237-268, vol.4.

[119] N. B. Hadj-Alouane, S. Lafortune, and L. Feng, "Variable lookahead supervisory control with state information," *IEEE Transactions on Automatic Control,* 1994, pp. 2398-2410, vol.39.

[120] M. Heymann and F. Lin, "On-line control of partially observed discrete event systems," *Discrete Event Dynamic Systems,* 1994, pp. 221-236, vol.4.

[121] R. Kumar and V. K. Garg, "Control of stochastic discrete event systems modeled by probabilistic languages," *IEEE Transactions on Automatic Control,* 2001, pp. 593-606, vol.46.

[122] F. Lin, "Robust and adaptive supervisory control of discrete event systems," *IEEE Transactions on Automatic Control,* 1993, pp. 1848-1852, vol.38.

[123] L. Grigorov and K. Rudie, "Near-optimal online control of dynamic discrete-event systems," *Discrete Event Dynamic Systems,* 2006, pp. 419-449, vol.16.

[124] C. Yi-Liang, S. Laortune, and L. Feng, "How to reuse supervisors when discrete event system models evolve," in *Proceedings of the 36th IEEE Conference on Decision and Control.*, 1997, pp. 2964-2969 vol.3.

[125] D. Gordon and K. Kiriakidis, "Adaptive supervisory control of interconnected discrete event systems," in *Proceedings of the 2000 IEEE International Conference on Control Applications*, 2000, pp. 935-940.

[126] D. Gordon-Spears and K. Kiriakidis, "Reconfigurable robot teams: modeling and supervisory control," *IEEE Transactions on Control Systems Technology,* 2004, pp. 763-769, vol.12.

[127] K. Kiriakidis and D. Gordon, "Supervision of multiple-robot systems," in *Proceedings of the 2001 American Control Conference.*, 2001, pp. 2117-2120, vol.3.

[128] W. Xi, P. Lee, A. Ray, and S. A. P. S. Phopa, "A behavior-based collaborative multi-agent system," in *IEEE International Conference on Systems, Man and Cybernetics*, 2003, pp. 4242-4248, vol.5.

[129] K. Jamie, R. K. Pretty, and R. G. Gosine, "Coordinated execution of tasks in a multiagent environment," *IEEE Transactions on Systems, Man and Cybernetics, Part A,* 2003, pp. 615-619, vol.33.

[130] S. Kimura, M. Takahashi, T. Okuyama, S. A. T. S. Tsuchiya, and Y. A. S. Y. Suzuki, "A fault-tolerant control algorithm having a decentralized autonomous architecture for space hyper-redundant manipulators," *IEEE Transactions on Systems, Man and Cybernetics, Part A,* 1998, pp. 521-527, vol.28.

[131] C. G. Cassandras and S. Lafortune, *Introduction to discrete event systems*, Second Edition ed. Norwell, Massachesetts, USA: Kluwer Academic Publishers, 1999.

[132] E. H. Mamdani, "Advances in the linguistic synthesis of fuzzy controllers," *International Journal of Man-Machine Studies,* 1976, pp. 245-254, vol.8.

[133] R. Bellman, "On the Theory of Dynamic Programming," in *Proceedings of the National Academy of Sciences*, 1952.

[134] V. D. Blondel and J. N. Tsitsiklis, "A survey of computational complexity results in systems and control," *Automatica,* 2000, pp. 1249-1274, vol.36.

[135] A. Tsalatsanis, K. Valavanis, and N. Tsourveloudis, "Mobile robot navigation using sonar and range measurements from uncalibrated cameras," *Journal of Intelligent and Robotic Research Systems,* 2007, pp. 253-284, vol.48.

[136] A. Tsalatsanis, K. Valavanis, and A. Yalcin, "Vision based target tracking and collision avoidance for mobile robots," in *International Conference on Artificial Intelligence*, 2006, pp. 652-658, vol.2.

[137] A. Tsalatsanis, K. Valavanis, and A. Yalcin, "Vision based target tracking and collision avoidance for mobile robots," *Journal of Intelligent and Robotic Research Systems,* 2008, pp. 285-304, vol.48.

[138] A. Tsalatsanis, K. P. Valavanis, A. Kandel, and A. Yalcin, "Multiple sensor based UGV localization using fuzzy extended Kalman filtering," in *Mediterranean Conference on Control & Automation*, 2007, pp. 1-8.

[139] A. Tsalatsanis, A. Yalcin, and K. Valavanis, "Dynamic task allocation in cooperative robot teams: A limited lookahead control policy," *submitted at IEEE Transactions on Automation Science and Engineering,* 2008.

[140] A. Tsalatsanis, A. Yalcin, and K. P. Valavanis, "Automata-based supervisory controller for a mobile robot team," in *IEEE 3rd Latin American Robotics Symposium, 2006. LARS '06.*, 2006, pp. 53-59.

Appendices

Appendix A HSI Color Space

The hue, H, of a color is determined by its angle with respect to the red axis [16]. $H=60^o$ corresponds to yellow. Thus, a threshold on the hue component in the area of 55^o to 65^o can easily extract yellow pixels from a color image. Additionally, saturation S of a color is the degree to which the color is undiluted by white [16]. Saturation's range for yellow obstacles has been experimentally determined to be within *(0.35, 0.43),* where *0* corresponds to unsaturated and *1* to fully saturated colors. Applying both hue and saturation thresholds in a HSI color space image results in a new image in which yellow pixels are represented as white according to the equation:

$$g(x,y) = \begin{cases} 255, & 55^o \le H(x,y) \le 65^o \text{ and } 0.35 < S(x,y) < 0.43 \\ 0 & \text{else} \end{cases}$$

Fig. A.1 shows a color image containing a yellow obstacle and the corresponding output after thresholding.

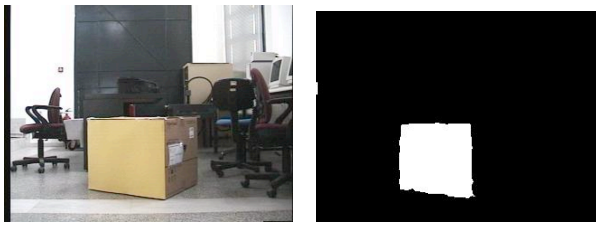
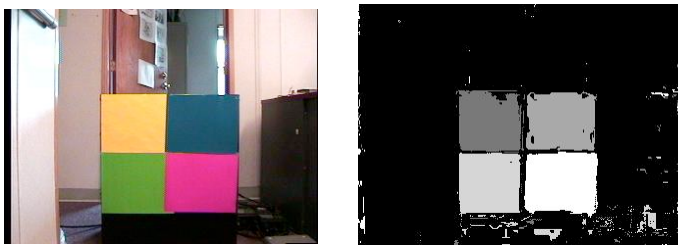


Fig. A.1 Applying threshold technique in HSI color space

The HSI color space may also be used to extract red, blue and green colors. The hue's value for red color is $H = 0^o \; or \; H = 360^o$, while the saturation's range has been determined to be within *(0.70, 0.80)*. Similarly, the hue's value for green color is $H = 120^o$, while its saturation varies between *(0.35, 0.54)*. Finally, blue color has a value of hue $H = 240^o$, and saturation between *(0.66, 0.94)*. Fig. A.2 presents a color image containing the four color obstacle (a) and the result of the threshold technique (b) according to the equation:

$$g(x,y) = \begin{cases} 255, & 355^o \le H(x,y) \le 5^o \; and \; 0.70 \le S(x,y) \le 0.80 \\ 200, & 115^o \le H(x,y) \le 125^o \; and \; 0.35 \le S(x,y) \le 0.54 \\ 150, & 235^o H(x,y) \le 245^o \; and \; 0.66 \le S(x,y) \le 0.94 \\ 100, & 55^o H(x,y) \le 65^o \; and \; 0.35 \le S(x,y) \le 0.43 \\ 0, & else \end{cases}$$



(a) (b)

Fig. A.2 Thresholding to extract red, green, blue and yellow obstacles using the HSI color space

Appendix B CMYK Color Space

The CMYK color space represents each color in its secondary spectral components cyan, magenta, yellow. One of its primary components is *yellow*. Pixels of a yellow obstacle refer to pixels with highest intensity in the yellow's component image that can be extracted from its histogram. Additionally, yellow color pixels are presented as black on the *cyan* component. A new image may be created by applying a threshold technique to the CMYK image according to:

$$g(x,y) = \begin{cases} f(x,y) & , Y(x,y) > T_1 \text{ and } C(x,y) < 5 \\ 0 & \text{else} \end{cases}$$

T_1 is the highest pick in yellow's component histogram minus a tolerance of *10%*, *Y(x,y)* and *C(x,y)* the intensity functions of *Yellow* and *Cyan* respectively. Fig. A.3 shows a color image contains a yellow obstacle (a), the thresholded image (b) and the histogram of the yellow component (c).

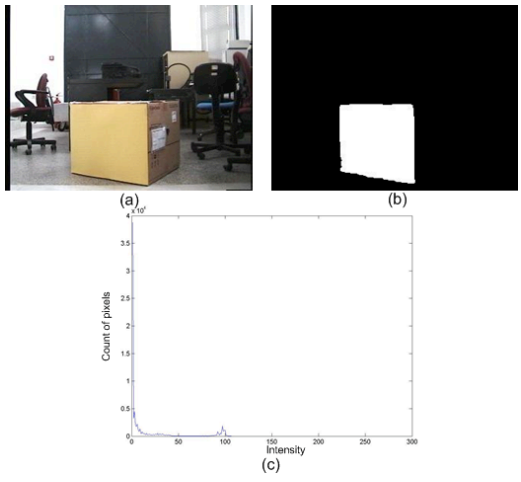


Fig. A.3 Applying threshold technique in CMYK color space.

The CMYK color space may also be used to extract red, green and blue colors from a color image. Red color corresponds to the highest values on *magenta* component and its *cyan* values varry between *(80, 115)*. Green color corresponds to high values on the *yellow* component and its *magenta* values varry between *(80, 135)*. Finally, blue color corresponds to the highest values on the *cyan* component and at the same time its magenta values varry between *(180, 230)*. Fig. A.4, presents a color image containing the four color obstacle (a), the result of the threshold technique (b) according to equation (7) and the histograms of cyan (c), magenta (d) and yellow (e) components.

$$g(x,y) = \begin{cases} 255, & M(x,y) > T_3 \; and \; 80 \le C(x,y) \le 115 \\ 200, & Y(x,y) \ge T_1 \; and \; 80 \le M(x,y) \le 135 \\ 150, & C(x,y) \ge T_2 \; and \; 180 \le M(x,y) \le 230 \\ 100, & Y(y,x) \ge T_1 \; and \; C(x,y) < 5 \\ 0, & else \end{cases}$$

T_1, T_2 and T_3 are the highest picks with the highest intensity values in yellow's, cyan's and magenta's component histograms respectively, minus a tolerance of *10%*. *C(x,y)*, *M(x,y)* and *Y(x,y)* are the intensity functions of the *Cyan*, *Magenta* and *Yellow* component respectively.

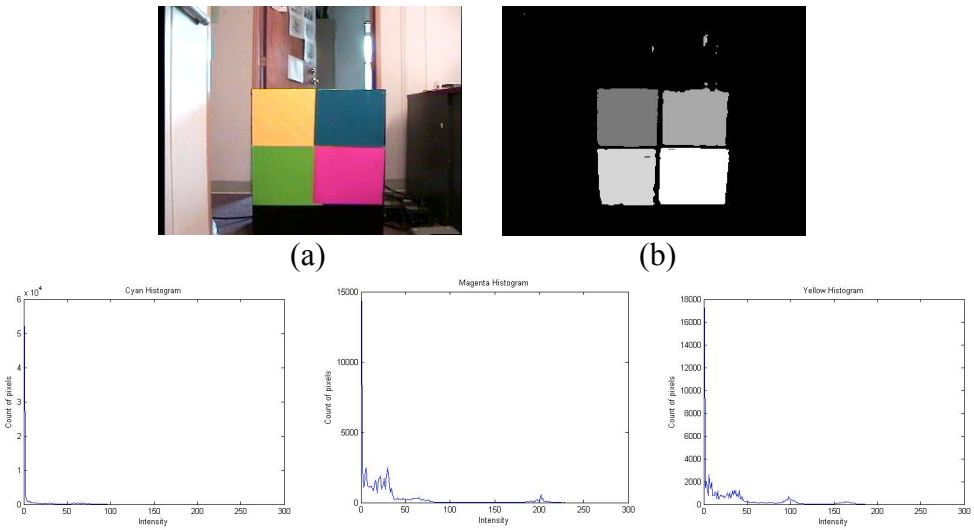


Fig. A.4 Thresholding to extract red, green, blue and yellow obstacles using the CMYK color space

Appendix C Color Segmentation Results

Fig. A.4 illustrates a multicolor obstacle located at different distances from the robot, under different lighting conditions (a), and the result of the threshold technique applied to the YC_bC_r (b), HSI (c), and CMYK (d) color spaces.

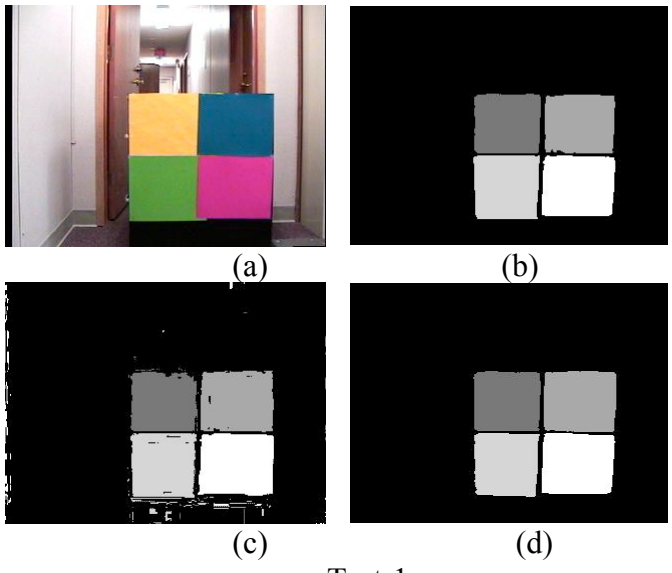
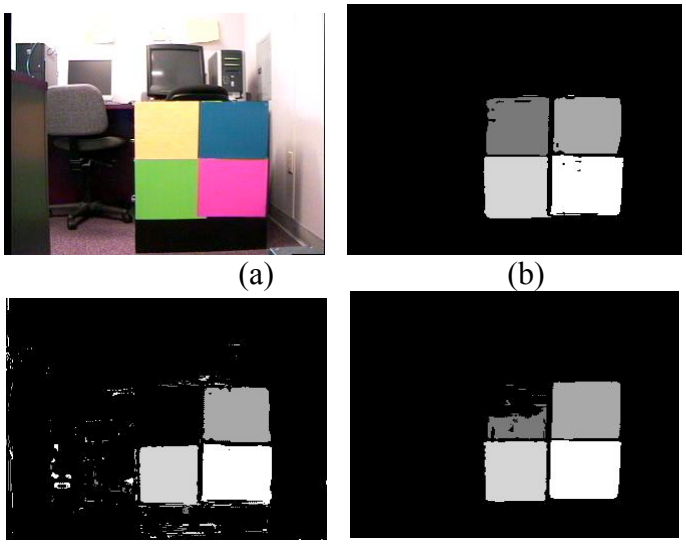


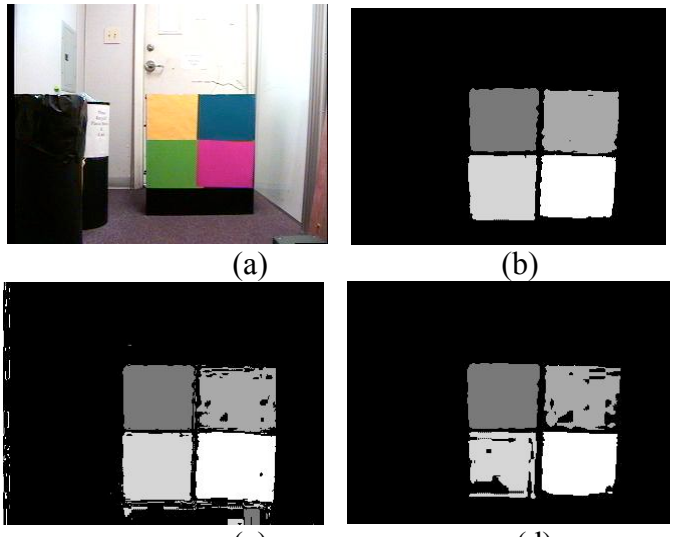
Fig. A.5 Thresholding to extract red, green, blue and yellow obstacles using the three color spaces



(a) (b)

(c) (d)

Test-3

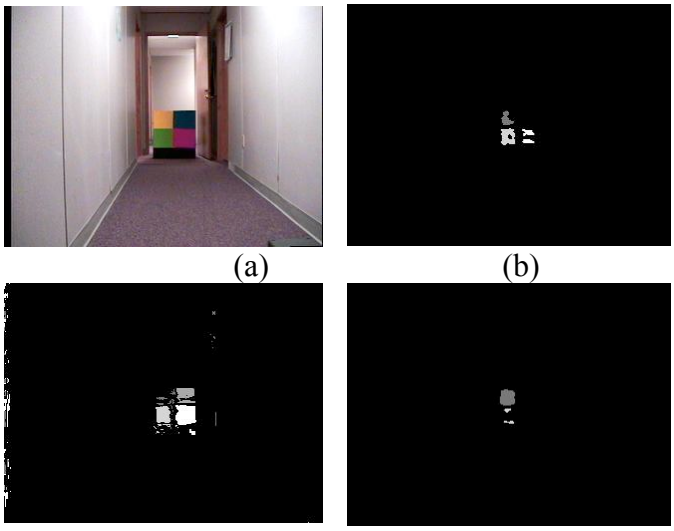


(a) (b)

(c) (d)

Test-4

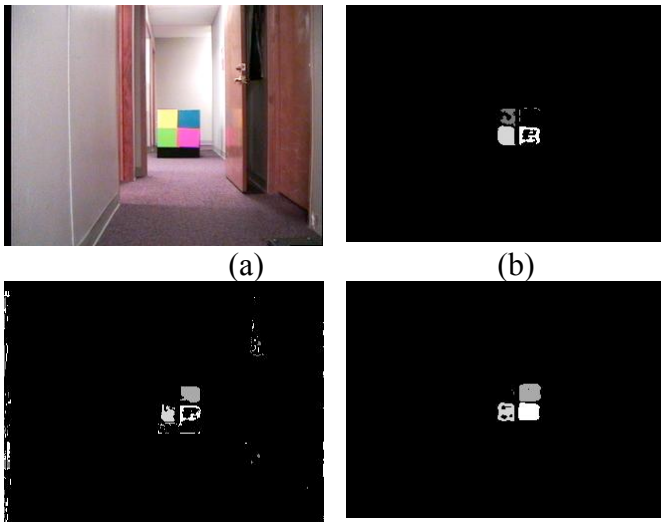
Fig. A.5 (Continued)



(a) (b)

(c) (d)

Test-5

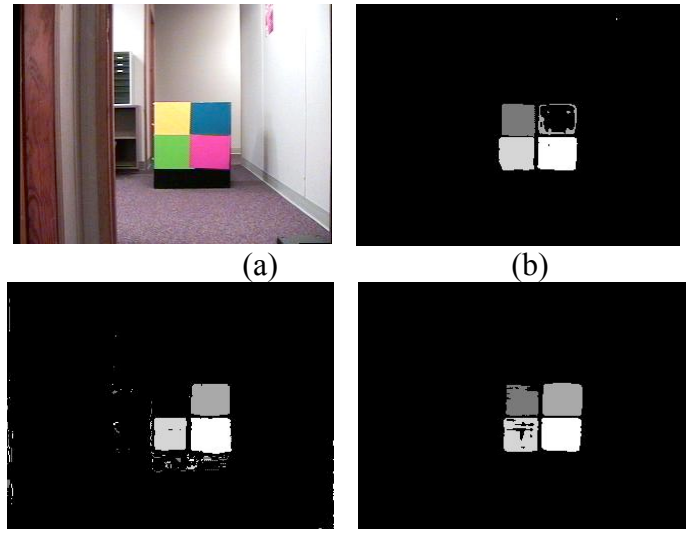


(a) (b)

(c) (d)

Test-6

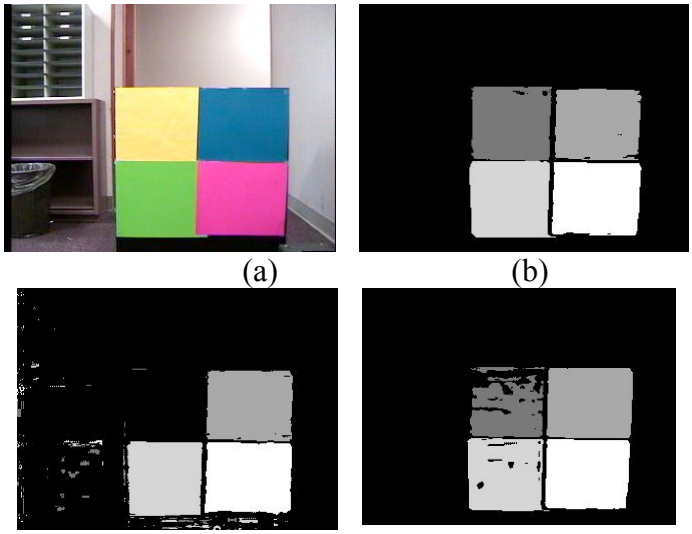
Fig. A.5 (Continued)



(a) (b)

(c) (d)

Test-7

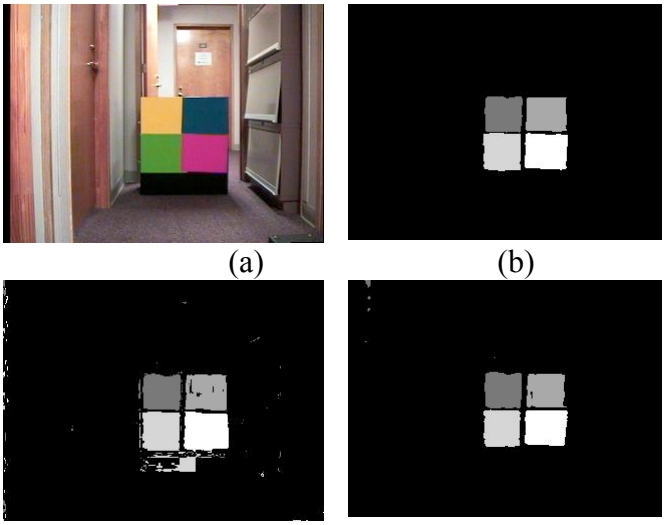


(a) (b)

(c) (d)

Test-8

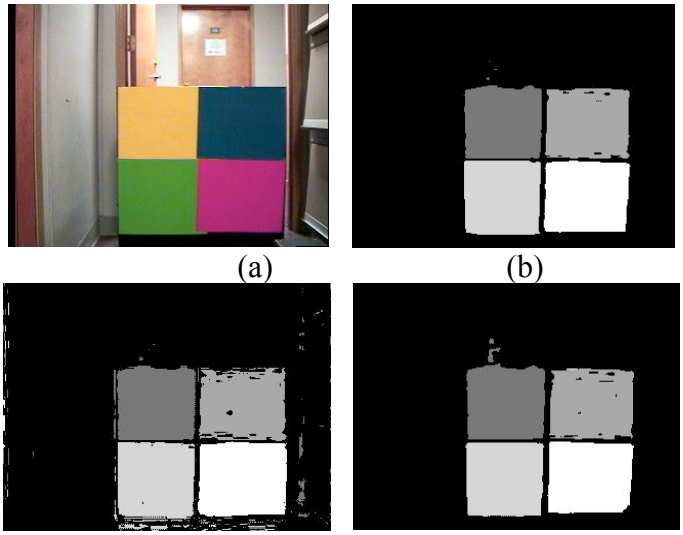
Fig. A.5 (Continued)



(a) (b)

(c) (d)

Test-9



(a) (b)

(c) (d)

Test-10

Fig. A.5 (Continued)

As observed, the threshold technique using the YC_bC_r color space performs better in identifying all colors, while background pixels have been considerably suppressed.

Finally, Fig. A.6 shows a more realistic and complicated scene with yellow objects that demonstrate the ability of the threshold technique as applied to the YC_bC_r color space to identify objects of different size and orientation.

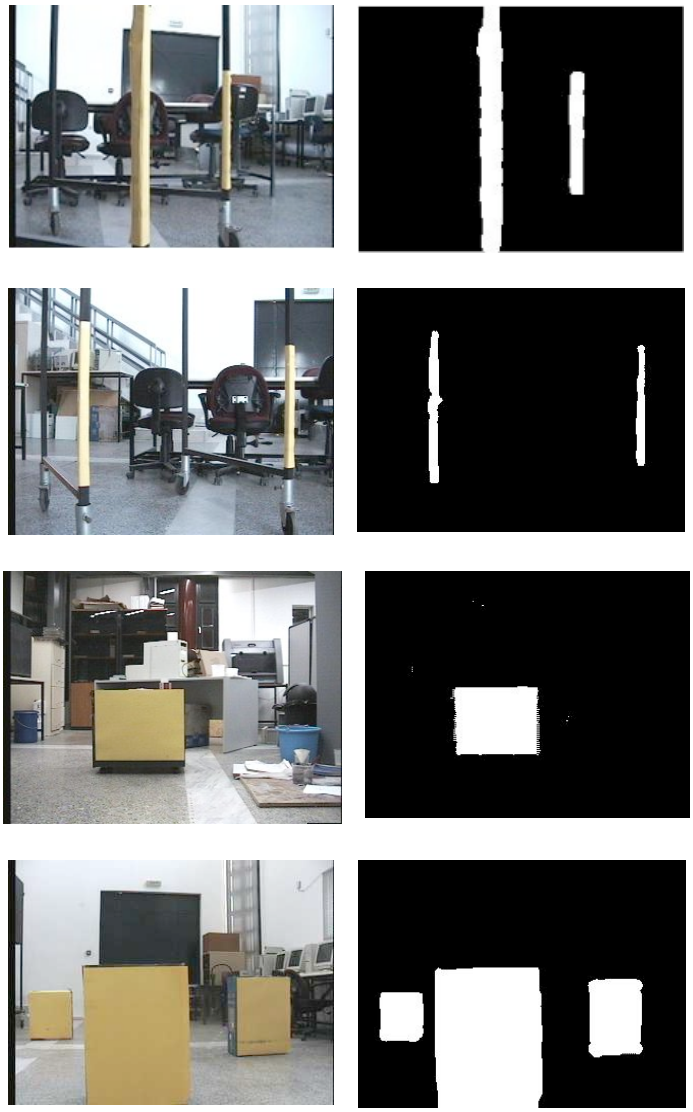


Fig. A.6 Applying threshold technique to extract yellow obstacles using the YC_bC_r color space

About the Author

Athanasios Tsalatsanis received his Diploma in Production Engineering and Management and his Masters in Production Engineering from Technical University of Crete, Greece in 2001 and 2003 respectively. As a doctoral student at the University of South Florida in the Department of Industrial and Management Systems Engineering, he worked in several programs and published several articles related to autonomous robot teams. His research interests include soft computing, control of autonomous systems and information systems.