DEPARTAMENT D'ENGINYERIA I CIÈNCIES DELS COMPUTADORS

UNIVERSITAT JAUME I

Agent-Based Architecture for Multirobot Cooperative Tasks: Design and Applications

PH. D. THESIS

Presented by:

PATRICIO NEBOT ROGLÁ

Supervised by:

ENRIC CERVERA MATEU
ÁNGEL PASCUAL DEL POBIL Y FERRÉ

Castelló, 2007

DEPARTAMENT D'ENGINYERIA I CIÈNCIES DELS COMPUTADORS

UNIVERSITAT JAUME I

Arquitectura basada en Agentes para Tareas Cooperativas Multirobot: Diseño y Aplicaciones

TESIS DOCTORAL

Presentada por:

PATRICIO NEBOT ROGLÁ

Supervisada por:

ENRIC CERVERA MATEU
ÁNGEL PASCUAL DEL POBIL Y FERRÉ

Castelló, 2007

Pa Maria Jose i la meua família,
els dos pilars bàsics en la meua vida.
Pel vostre suport i la vostra paciència.
Sense vosatros açò no haguera sigut possible.

Abstract

This thesis focuses on the development of a system in which a team of heterogeneous mobile robots can cooperate to perform a wide range of tasks. In order that a group of heterogeneous robots can cooperate among them, one of the most important parts to develop is the creation of an architecture which gives support for the cooperation. This architecture is developed by means of embedding agents and interfacing agent code with native low-level code. It also addresses the implementation of resource sharing among the whole group of robots, that is, the robots can borrow capabilities from each-other.

In order to validate this architecture, that is to check if it is suitable for implementing cooperative tasks, some cooperative applications have been implemented. The first one is an application where a group of robots must cooperate in order to safely navigate through an unknown environment. In this case, only one robot has a camera mounted on it and the none of the other robots have any type of sensor. The robot with the camera calculates the optical flow values from the images that it gets from the camera, and from these values calculates the "time to contact" values that determine the time that the robot has before colliding with an obstacle in front of it. This information is shared among the team so that any robot can navigate without colliding with the obstacles.

The second cooperative application implemented in order to validate the architecture consists of enabling the team of robots to create a certain formation and navigate maintaining this formation. Four robots have been used in this case, one robot with a laser system, another one with a camera, and finally, another two robots without any type of sensor. The application consists of two parts or stages. The first one is the creation of the formation, where the robot with the camera, called leader, can detect where the rest of the robots are in the environment and indicates to them which is their initial position in the formation. In the second stage, once the formation has been created, the robots must be able to navigate through an environment following the path that the robot with the laser, called conductor, indicates. At the beginning, it was desired that the robots could follow the conductor only using the information that this robot provides to the rest of the group, but this approach had to be abandoned due to the odometry errors of the robots. To solve this problem, the camera of the leader is used so that robots which lose their correct position in the formation can re-align themselves.

Finally, in an attempt to facilitate access to the robots of the team and to the information that their accessories provide, a system for the teleoperation of the team has been implemented. This system can be used for teaching robotics or to facilitate the tasks of programming and debugging in the research tasks.

Some of the main contributions of the thesis include:

- Design and implementation of a control agent-based architecture for the cooperation of a team of heterogeneous mobile robots.

- Implementation of a cooperative task for the navigation between several robots by means of the use of the optical flow and time to contact techniques, where the most interesting thing is how the robots cooperate to navigate in a safe way.
- Development of a cooperative task for the creation and maintenance of multirobot formations. The robots must cooperate in order to follow the path indicated by the conductor robot without losing the formation. It has been necessary to develop new methods in order to achieve this end:
  - A new method for the visual localization of the robots; using a very common and simple target it is possible to localize one robot and determine its position and orientation relative to the robot with the camera.
  - An improvement in the calculation of the path to be followed by the formation; this method tries to soften the sharp angles that are present in the original path calculated by the navigation software of the robots.
  - A new design of formation navigation based on a "conductor-reference" approach, with a series of virtual points at a distance from the conductor position for the construction of the formation, and Bezier curves for the movement of each robot.
  - Implementation of a cooperative method in order to eliminate the odometry errors of the robots and exploit the resources of each robot in order to benefit the whole team. The cooperation is done in the way that the robots without sensors can use the resources of the others to navigate while maintaining the formation.
- Implementation of a teleoperated system to facilitate access to all the robots of the team and all their accessories at a glance. This system helps the learning of the robots and is an important tool to detect failures in the development of cooperative tasks.

Resumen

Motivación

Esta tesis se centra en conseguir que un equipo de robots, cada uno de ellos con diferentes características tanto de diseño físico como lógico, sean capaces de cooperar y colaborar entre ellos y a su vez con humanos para la realización de tareas de diversa índole.

Establecer estos mecanismos de cooperación entre los robots implica considerar un problema de diseño del comportamiento cooperativo, es decir, dado un grupo de robots, un entorno y una tarea a realizar, ¿cómo deber llevarse a cabo la cooperación? Tales problemas implican varios desafíos, pero entre todos ellos el más importante es la definición de la arquitectura que da soporte a la cooperación.

En líneas generales, las arquitecturas previas tienden a considerar a cada uno de los robots como un todo y se centran en la implementación de la cooperación a nivel de ente. La idea principal seguida para la implementación de la arquitectura descrita en esta tesis es que cada robot puede usar también las funcionalidades del resto de robots del equipo para compensar la falta de esta funcionalidad o para poder mejorar alguna ya disponible, generando una cooperación a un nivel más bajo.

Además, con la intención de validar esta arquitectura, varias aplicaciones cooperativas han sido desarrolladas. En la primera de ellas, un grupo de robots debe cooperar para lograr una navegación segura de todo el grupo a través de un entorno desconocido. La segunda aplicación consiste en hacer posible que un equipo de robots pueda crear una determinada formación y mantenerla mientras navega por el entorno. Finalmente, un sistema de teleoperación ha sido implementado en un intento de facilitar el aprendizaje del manejo de los robots a usuarios inexpertos, así como ayudar en las tareas de depurado de las aplicaciones que se creen.

Objetivos

El principal objetivo de esta tesis es la definición e implementación de una arquitectura de control que dé soporte para la cooperación y colaboración entre un equipo heterogéneo de robots móviles, y que además facilite la implementación de tareas cooperativas para dicho equipo.

A parte de este objetivo principal, se incluye otros objetivos parciales:

- Implementación y diseño de la arquitectura haciendo posible la compartición de los recursos de un robot entre el resto de robots del equipo.
- Definición de la arquitectura siguiendo criterios como reusabilidad del software, escalabilidad y facilidad de uso.
- Definición de la arquitectura tratando de separar la parte de acceso a los recursos físicos del robot de la parte más comportamental, por medio de un middleware que separa estas dos capas.

- Implementación de la arquitectura siguiendo un enfoque basado en agentes.
- Acceso a los elementos físicos del robot y a sus funcionalidades de forma transparente para el programador.

El objetivo principal para las dos aplicaciones creadas para demostrar que la arquitectura es válida para la funcionalidad con que fue pensada es mostrar que por medio de esta arquitectura un equipo de robots, con diferentes habilidades de cada uno de sus miembros, es capaz de colaborar y cooperar para ayudar a los miembros más desfavorecidos a completar la tarea requerida.

Por ultimo, el objetivo marcado para el sistema de teleopereción es el facilitar el acceso al manejo y a la información de los robots del equipo para ayudar en tareas de aprendizaje para estudiantes y en tares de desarrollo y depuración de tareas para el equipo.

Metodología

Esta tesis se centra en el desarrollo de un sistema en el cual un equipo de robots móviles heterogéneos pueda cooperar para realizar una gran variedad de tareas. Para que este grupo de robots heterogéneos pueda cooperar, una de las partes más importantes a desarrollar es la creación de una arquitectura que de soporte para esa cooperación. Esta arquitectura ha sido desarrollada mediante el uso de agentes embebidos y código nativo de bajo nivel con interfaz para código agente. Además implementa la compartición de recursos entre el grupo de robots, esto es, los robots pueden compartir capacidades o habilidades unos con otros.

Para la comprobación de la arquitectura, se han implementado dos aplicaciones. En la primera de ellas, un grupo de robots debe cooperar para navegar de forma segura a través de un entorno desconocido. En este grupo, solo un robot dispone de una cámara montada sobre el que le permite reconocer el entorno; el resto de robots son completamente ciegos, es decir, no disponen de sensores que les ofrezcan información sobre el entorno. El robot con la cámara calcula los valores del flujo óptico a partir de las imágenes, y a partir de estos valores calcule los valores para el "tiempo de contacto", es decir, el tiempo que el robot dispone antes de chocar con algún obstáculo frente a el. Esta información es compartida entre el equipo para que cada robot pueda navegar sin colisionar con los obstáculos.

La segunda aplicación consiste en hacer posible que un equipo de robots pueda crear una determinada formación y mantenerla mientras navega por el entorno. En este caso han sido usados 4 robots, cada uno de ellos con diferentes características. Uno dispone de un sistema láser, otro de cámara, y el resto sin ningún tipo de sensor. La aplicación está compuesta por dos fases. En la primera de ellas se crea la formación de forma totalmente autónoma. El robot con la cámara, llamado "leader", detecta al resto de robots en el entorno y les indica cual es la posición que deben ocupar en la formación. En la segunda fase, una vez la formación ha sido creada, los robots deben ser capaces de navegar a través del entorno siguiendo una determinada trayectoria indicada por el robot con láser, llamado "conductor". En una primera instancia, el deseo fue que los robots pudieran seguir al conductor usando únicamente la información que este robot sirve a todo el grupo, pero debido a errores de odometría se tuvo que desechar este enfoque. Para solucionar este problema, la cámara del leader es usada para recolocar a los robots en la formación en el caso de que ellos hayan perdido su posición.

Finalmente, con la intención de facilitar el acceso a los robots y a la información que sirven sus accesorios, se ha implementado un sistema para la teleoperación del equipo. Este sistema podría ser usado para enseñar los fundamentos de la robótica a estudiantes o para facilitar las tareas de programación y depurado en el ámbito de la investigación. En este sistema, dos subsistemas han sido tenidos en cuenta, por un lado el modelo de control y por otro lado el modelo de interface. En referencia al modelo de control, un modelo supervisor ha sido escogido para su implementación, donde es posible tanto actuar directamente sobre los robots como simplemente permanecer en un estado pasivo y supervisar la operación de los robots. Para el modelo de interface, un modelo multimodal, donde diferentes tipos de información pueden ser mostrados al mismo tiempo ayudando a la rápida comprensión del entorno del robot y facilitando así la toma de decisiones por parte del operador en caso de manejar directamente los robots.

Aportaciones

Entre las principales contribuciones de esta tesis, cabe destacar:

- Diseño e implementación de una arquitectura de control basada en agentes para la cooperación de un equipo de robots móviles heterogéneos.
- Implementación de una tarea cooperativa para la navegación entre varios robots por medio del uso de las técnicas de flujo óptico y tiempo de contacto, donde la aportación más interesante es como cooperan los robots para navegar de una forma segura.
- Desarrollo de una tarea cooperativa para la creación y mantenimiento de formaciones de múltiples robots móviles. Los robots deben cooperar para seguir el camino indicado por el robot conductor sin perder la figura de la formación. En el desarrollo de esta aplicación ha sido necesaria la implementación de algunos métodos nuevos:
  - Un nuevo método para la localización visual de los robots; usando un objetivo muy común y simple es posible localizar un robot y determinar su posición y orientación respecto al robot con cámara.
  - Perfeccionamiento en el calculo del camino a seguir por la formación; este método trata de suavizar los ángulos bruscos que hay presentes en el camino original calculado por el software de navegación de los robots.
  - Un nuevo diseño de navegación de la formación basado en un enfoque “referencia al conductor”, con una serie de puntos virtuales desplazados sobre la posición del conductor para la construcción de la formación, y curvas de Bezier para el movimiento particular de cada robot.
  - Implementación de un método cooperativo para eliminar los errores de odometría de los robots y explotar los recursos de cada robot en beneficio de todo el equipo. La cooperación está enfocada a que los robots sin sensores puedan usar los recursos de los otros robots para navegar manteniendo la formación.
- Implementación de un sistema teleoperado para facilitar el acceso a los robots del equipo y a todos sus accesorios de un vistazo. Este sistema facilita el aprendizaje de los robots y es una importante herramienta para la detección de fallos en el desarrollo de tareas cooperativas.

Conclusiones y trabajo futuro

Esta tesis considera el problema de tener varios robots trabajando juntos pare el desarrollo de tareas comunes, es decir, cooperar entre ello en la resolución de la tarea. Uno de los puntos más importantes para conseguir la cooperación en un sistema es el desarrollo de una arquitectura que de soporte a esa cooperación. Por tanto, esta tesis propone la creación de una arquitectura que permita el desarrollo de tareas cooperativas con un equipo de robots heterogéneos.

Los resultados obtenidos con las aplicaciones implementadas demuestran que la arquitectura generada es válida para dar soporte a cualquier tares que requiera cooperación entre los robots. Además, la arquitectura da soporte para que todos aquellos equipos de robots con un limitado poder de sensorización puedan realizar tareas por medio de la compartición de recursos entre el equipo, proporcionando a los robots con menos poder de sensorización con la capacidad de usar o aprovecharse de los recursos de los otros robots en su propio beneficio y por tanto de todo el grupo.

Finalmente, como puede verse a partir de varios trabajos presentados en la tesis, el sistema de teleoperación desarrollado puede ser usado en tareas de enseñanza en robótica o para facilitar las tareas de programación y depurado en las tareas de investigación.

Respecto al trabajo futuro, en esta tesis se propone las siguientes líneas:

- En relación con la arquitectura de control: mejorar el protocolo de interacción creado para dar acceso a los agentes a un elemento determinado. En este protocolo, cuando un agente gana el acceso al elemento físico en cuestión, tiene ese acceso hasta que finaliza su tarea. Una mejora seria introducir un tiempo máximo para la realización de la tarea y una vez este tiempo ha finalizado, el agente debe esperar hasta conseguir otra vez el permiso. Otra posibilidad podría ser añadir un sistema de prioridades, así si un agente tiene una tarea urgente que realizar, puede conseguir inmediatamente el acceso, echando temporalmente al agente que en ese momento estaba usando el elemento. O incluso, una mezcla de ambos enfoques podría ser más beneficioso.

- En relación con la aplicación del flujo óptico:

  – Uno de los problemas con la técnica del flujo óptico es que es imposible distinguir las situaciones en que un objeto está demasiado cerca al robot. En este caso los sonars pueden ser usados para distinguir estas situaciones y permitir al robot realizar una maniobra evasiva.

  – Otro problema es el calculo del flujo óptico en superficies homogéneas. Nuevas técnicas o formas de detectar estas situaciones deben ser estudiadas y mejorar el sistema para poder tratar con ello.

  – También es importante tratar de reducir el tiempo que se consume en el cálculo del flujo óptico. Para ello, nuevo métodos más rápidos que el usado en esta tesis pueden ser usados para conseguir que el robot pueda reaccionar más rápido a cambios en el entorno.

- En relación con las formaciones multirobot:

  – La localización de los robots por medio de los objetivos de color ha sido una tarea muy dura debido a la sensibilidad del sistema de visión a las condiciones de iluminación. Trabajo podría ser realizado en el desarrollo de procedimientos mas robustos de localización usando combinaciones de diferentes características para detectar el objetivo en el espacio de la imagen.

  - Uno de los puntos todavía no desarrollados corresponde con el cambio de formación mientras los robots se mueven. Esto puede ser útil para evitar obstáculos o pasar a través de pasajes estrechos donde la formación no cabe. Con la actual implementación no es muy difícil realizar esta tarea con la formación en movimiento, las curvas de Bezier pueden ser usadas para mover los robots a puntos virtuales que determinen la posición que los robots deben ocupara en la formación después del reposicionamiento.
  - Otro punto todavía no desarrollado es la detección de obstáculos en el camino de la formación. En este momento, todos los obstáculos estáticos son detectados y evitados por medio de la generación del camino que los robots deben seguir, pero los objetos dinámicos no pueden ser detectados. Es necesario estudiar como es posible, con los sensores disponibles, detectar obstáculos dinámicos y evitarlos. También es importante estudiar como evitar los obstáculos, esto es, si es necesario modificar la formación o es posible mover la formación como todo un bloque para evitar el obstáculo.

- En relación con el sistema de teleoperación: podría ser provechoso mejorar el sistema teleoperado incluyendo técnicas para facilitar al usuario la percepción del entorno. Para ello, técnicas para crear una representación virtual en 3D del entorno podrían ser implementadas. Además, es necesario incluir nuevas formas de introducir órdenes para los robots en un nivel superior y más cercano al operador. Finalmente, ahora las órdenes solo se realizan en un robot cada vez, por tanto, es necesario introducir mecanismos que faciliten la especificación de órdenes para todo el equipo o un subconjunto del mismo.

Acknowledgments

This thesis has been developed at the Robotic Intelligence Laboratory of the University Jaume I. First of all, I am most grateful to my advisors, Enric Cervera and Ángel Pascual del Pobil for their guidance and support during the development of this thesis. They provided extremely valuable help with the many difficulties that arose during this period. I have learned a great deal after all these years of working with them.

I would also like to thank the members of this laboratory, current and former, for their warm welcome and their help and encouragement from the beginning. I am specially grateful to Dániel Gómez, with whom I shared many hours of work in the early stages of my PhD. An important part of the work on the design and implementation of the architecture presented in this thesis is the result of earlier cooperation with him. I would also like to specially thank Raúl Pérez for his valuable help in the implementation of some of the visual tools added to the architecture. I would also like to specially thank Mario Prats, who has been always there when I have needed him to solve problems in Linux and its configurations. At end, thanks to Gabriel, Eris, Ester, Toni, Wirz, Pablo, Juan Carlos, Xabi, ... for their friendship.

Many other people have helped me in the implementation of the programs described in this thesis, some of them working on projects from which I was advisor. Benjamin Berton, from the *Ecole Centrale de Nantes* (France), and Gael Saintluc, from the *ISEN Toulon* (France), worked on an initial implementation of the architecture for a mobile manipulator during their stay in the Robotic Intelligence Laboratory. They worked on the extension of the architecture designed to allow the use of an arm mounted on a mobile basis proposed in section 3, and the implementation of a controller for the stereo camera mounted on the arm. Paras Mehta and Indramohan Shrivas, from the *Indian Institute of Technology Kanpur* (India), provided the initial implementation of the optical flow technique and the safe navigation using it in a single robot application explained in section 4. Siou Y Chiem, from the *École Centrale de Nantes* (France), Vincent Robin , from the *IFMA* (France), and Pierre Renaud, from the *LIRMM* (France), who provided the first ideas of the multirobot formations using vision-based control. They worked on a first implementation of follow-the-leader application, whose ideas I used in the implementation of the multirobot formations described in section 5. Finally, I would also like to specially thank Pedro J. Sanz and Raül Marín for using their valuable time to introduce me to the field of telerobotics and telemanipulation. Their advice was very valuable in the implementation of the teleoperation system in section 6.

During these years at the university I have had the privilege of carrying out part of my research in cooperation with people from other universities. Among them, I would like to thank Chris Melhuish, from the *University of West of England* (Bristol, England), for providing me the invitation for my first stay in a university of a foreign country. It was during this stay when I began to work with robot formations. There I also received help from other members of the laboratory, such as Alan Winfield, Ioannis Ieropoulos, Jan Dyre Bjerknes, Peter Jaeckel, Wenguo Liu and Chris Bytheway.

I would also like to thank people like Sergio, Ricardo, Jorge, Nacho, and and many others that I met there for smoothing my first contact with a different country, making my stay in Bristol such an enjoyable experience, and specially, for their unforgettable friendship.

As part of my teaching work during this years, I had the opportunity to be the advisor of the projects developed by some students, which allowed me to test some ideas that I had no time during that period to fully develop by myself. In particular, Kino, who worked in the implementation of the cooperative multirobot navigation using the technique of optical flow, and Cristina, who worked in the implementation of the teleoperation system.

People from other laboratories in this university have also provided highly valuable help. I am specially thankful to Joaquín Torres, for his friendship and his extremely valuable help.

Thanks in general to the Department of Engineering and Computer Science of the University Jaume I for providing me framework for carrying on my research and teaching work, and to the national and international institutions that have provide the funds for making this research possible.

Thanks also to all the friends who have made my last years at the university so enjoyable. Among those already mentioned, thanks to Pau, Juan Carlos, “Van der Khul” team, and many others, though not mentioned by name, have been no less important to me.

I would also like to thank very specially two groups of people that have always helped me, not in research themes, but on a personal level. On the one hand, I would like to thank my friends Lucio, Ruben, Sento, Boro, Molina, Murillo, and many others that always believed in me. Sometimes the best support is more simple than a good idea for the research. On the other hand, I would like to thank the “Grup Scout Orleyl”, my other group of friends and part of my life, in special to Cesar, Bernardo, Castello, Nuño, ... And finally, I would to specially mention to Moliner, who took care of my special “thing” while I was not here. He is more than a friend, he is like a brother. Many thanks to all of you.

And last but not least, to my family and my girlfriend Maria Jose, for their capacity to listen, to advise and for their patience with me, and, above all, for giving me their unconditional support, regardless of what I was doing with this (for them) “strange” bussiness.

Patricio Nebot Roglá
Castelló, Octubre 2007

Contents

List of Figures

Chapter 1

Introduction

This section explains the motivation behind the elaboration of this thesis, the physical setup used, and the main goals achieved.

1.1 Motivation

Since the beginning of time, biological systems have shown a certain intelligent behaviour, the capacity to cooperate. The interaction between organisms such as birds, ants, termites, primates, fish or wolves has been studied for many years ago [135, 245, 352, 353]. Observations from biological systems have been used as the inspiration for developments in robotics such as subsumption architectures or to implement inter-robot communication for groups of robots [11, 56].

If individuals within a group want to cooperate with each other, each member must be able to communicate with the others. This communication can take place directly via an explicit communication channel or indirectly through changes in the environment that the other robots can sense [252]. Some biological cooperative systems are described in the following paragraphs.

Cooperation among *bacteria*, the simplest biological organism that exists, is as old as life on earth. Over millions of years, bacterias have existed, and they have exhibited a very primitive type of cooperation. Bacterias are able to organize themselves in hierarchical colonies and can communicate among themselves using sophisticated chemical signals [44, 181]. In figure 1.1, an example of bacteria colony is shown.

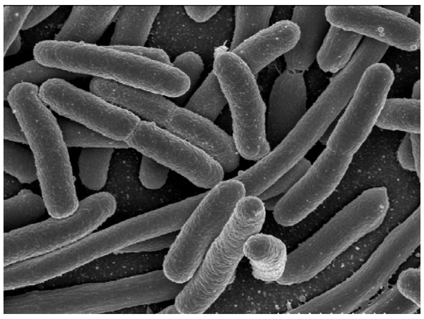


Figure 1.1: Bacteria organization in a colony.

These biochemical messages also are used by bacteria to exchange information among colonies of different species, or even other organisms. Collectively, bacteria can collect information from the environment, give a meaning to this information, develop common knowledge and learn from past experiences. In these cases, the colony behaves as a social community of high complexity.

Other societies that are able to cooperate are the insects, like ants, termites, bees or wasps [91, 366, 367]. The most representative are the *ants*, which exhibit a large number of cooperative behaviors. For example, when a worker ant discovers a new food source, it leaves a pheromone track during its return to the nest, as described in [293]. Recruited ants will follow this track to the food source with some variation while leaving their own pheromones. Any variation that results in a shorter track to the food, by chance, will be reinforced at a slightly faster rate. In that way, the ants can quickly establish the near optimal shortest path as an emergent consequence of a cooperative task, due to the fact that all the ants try to find the path. Figure 1.2 shows how ants follow the best path.

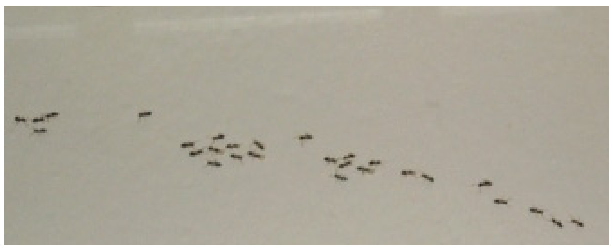


Figure 1.2: Ants searching the best path for returning to the nest.

In this case, the mechanism that ants use to communicate among them is the pheromone signals. The receiver of the communication can sense the pheromone at the location it was emitted. This mechanism, which is more complex than for bacteria, is still relatively simple. But, the ants also can communicate by signaling directly from antennae to antennae.

Another more complex example is the society that *wolves* form. Wolves are carnivores that form packs with strict social hierarchies and mating systems [335]. Wolves are territorial and mark its territory urinating on objects on the periphery and within territories. This is a communication system that resembles the ants and their chemical tracks. Moreover, wolves can communicate with pheromones excreted via glands near the anus and the dorsal surface of the tail.

Figure 1.3: Wolves cooperating in a hunt.

An example of cooperation is the fact that wolves hunt in packs, as can be seen in figure 1.3. During the hunt, each wolf cooperates by observing the actions of the others and the dominant male in particular. Each wolf knows all the members and can identify them individually, visually and by smell, and can communicate with any member. The communication consists of a combination of specific postures and vocalizations. These vocalizations and postures are learned during development in a social environment.

Other animals that show sophisticated cooperative behaviour are the *primates*. Higher primates are able to represent internal goals, plans, aptitudes and intentions of others and to construct collaborative plans jointly through social action [78].

In this case, the majority of interactions involve passive observation of collaborators via visual and auditory cues, which are interpreted as actions and intentions. Some explicit posing and gesturing is also utilized, which is used to establish and control ongoing cooperative interactions. Figure 1.4 shows a group of primates in a cooperative attitude.

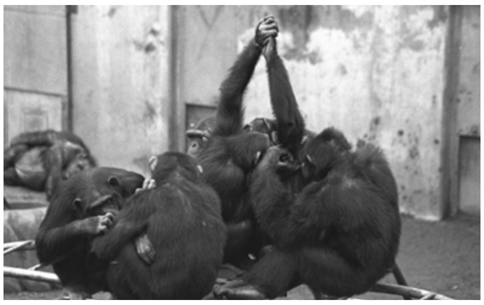


Figure 1.4: Primates in a cooperative attitude.

Finally, the most complicated organisms also show cooperative behaviours. *Humans* cooperate in many and varied ways. They exhibit cooperative relationships with mates, relatives, friends, organizations, and societies where they exchange resources for mutual benefit [95]. Moreover, humans can organize themselves into hierarchies and negotiate subdivision of tasks to perform large tasks, or adapt these subdivisions to changing circumstances while the task progresses.In figure 1.5, it is possible to see a group of humans cooperating in a task.

Figure 1.5: A group of humans cooperating in a task.

Humans make use of a broad set of communication systems. They make an extensive use of explicit communication, written and spoken. But they can also make use of implicit communication, such as posturing, known as body language, or even they can use explicit gesturing, like pointing or facial expressions.

But, why it is important to understand how biological systems cooperate? In this way, it is possible to determine what benefits are created for biological systems through cooperation and apply them in the design of cooperative robot systems. Knowing the cooperative mechanisms of biological systems makes it feasible to copy them in robot systems in order to solve similar problems, making the robots evolve the capabilities necessary for solving more complicated tasks. This subject of study has been called "Ethology".

Also, it is important to note the fact that to have a cooperative system it is necessary for more than one organism to have a relationship with another. So, to implement cooperation in a robotic system, it is necessary to have more than one robot working in the same environment, a multirobot system.

The advances in mobile robotics, computing power and wireless communications in the last years have made the development of communities of autonomous robots feasible. Lastly, there is an increasing interest in the development of systems of multiple autonomous robots, which are capable of exhibiting cooperative behaviour. This interest is due to the fact that having one single robot with multiple capabilities may be a waste of resources. Different robots, each one with its own configuration, are more flexible, robust and cost-effective. Moreover, the tasks to be performed may be too complex for one single robot, whereas they can be effectively done by multiple robots [13]. But unfortunately, there are also drawbacks, in particular regarding coordination and elimination of interference. So, it is important that the cooperation is done in a way that allows each robot to take advantage of the particular resources of the other robots in the system.

There are several applications in which having more than one robot working in parallel has improved the system's fault tolerance, and has reduced the time required for executing the tasks. Some of these applications are autonomous cleaning [179, 192], tour guiding in museums, art-galleries, or exhibitions [112], surveillance and patrolling [111, 359], rescue in disasters such as fires, floods, earthquakes [113], landmine detection [1] and autonomous exploration and mapping [65, 305].

This thesis takes this into account and aims to develop a system in which a team of heterogeneous mobile robots can cooperate to perform a wide range of tasks. In order that robots can cooperate among them, one of the most important parts to develop is the creation of an architecture which provides support for the cooperation. The implementation of this architecture is described in this thesis. Also, to validate this architecture, some cooperative tasks have been implemented and are depicted in this thesis. Moreover, at the end, an application is described which allows an easy communication and tele-operation with all the robots of the team and all their accessories.

The rest of this chapter is organized as follows. Section 1.2 provides an overall description of the problem considered in this thesis. Next, section 1.3 summarizes the main contributions of this work. Finally, section 1.4 outlines the contents of the following chapters of the thesis.

1.2 Problem description

As it has been said before, the object of this thesis is the creation of an architecture (Acromovi - an acronym in Spanish which stands for Cooperative Architecture for Intelligent Mobile Robots) that gives support for the cooperation of a team of multiple heterogeneous robots. And in order to validate this architecture some cooperative applications have been implemented.

The main goal is the design and implementation of a distributed architecture for the programming and control of a team of coordinated heterogeneous mobile robots, which are able to cooperate among themselves and with people in the accomplishment of tasks of service in daily environments. This architecture is a development by means of embedding agents and interfacing agent code with native low-level code. It also addresses the implementation of resource sharing among the whole group of robots, that is, the robots can borrow capabilities from each-other.

To validate the architecture, two cooperative applications or tasks have been implemented. The first one is an application where a group of robots must cooperate in order to safely navigate through an unknown environment. In this case, only one robot is equipped with a camera and the rest of robots do not have any type of exteroceptive sensors. The robot with the camera calculates the optical flow values from the images obtained from the camera, and from these values calculates the "time to contact" values that determine the time that the robot has before colliding with an obstacle in front of it. This information is shared among the team so that any robot can navigate without colliding with the obstacles.

The second cooperative application implemented in order to validate the architecture consists of the team of mobile robots creating a certain formation and navigating while maintaining this formation. In this case, once more, only one robot is equipped with a camera, another one includes a laser sensor and the rest of robots do not have any type of exteroceptive sensors. The robot with the laser is the one which indicates the path to follow, leading the whole team of robots. The robot with the camera has two tasks. In the first stage, it is in charge of detecting the rest of the robots in the environment and indicates to them which is their initial position in the formation. Then, while the formation is moving, it is in charge of detecting if the follower robots are in the correct position in the formation. Finally, the robots without sensors, at beginning, wait for the instructions to create the formation and then they have to follow the leader and modify its position if they are not in the correct one, this fact is indicated by the robot with camera.

Finally, since the robots can be online and can have access to Internet, the teleoperation systems, which enable multiple users to interact with the robots, have gained success and have become a very useful tool in teaching robotics. In an attempt to facilitate the access to the robots of the team and to the information that serves its accessories, a system for the tele-operation of the team has been implemented. This system can be used for teaching robotics or to facilitate the tasks of programming and debugging in the research tasks.

1.2.1 Physical setup

In order to understand the purpose and working environment of the architecture and the applications, their physical embedding instances are described in this section.

- **Pioneer-2 Robots.** For the development of the architecture, a team consisting of six Pioneer-2 mobile robots, manufactured by ActivMedia Robotics, has been used. Though sharing the same platform, the robots have different features, such as different accessories (cameras, laser, grippers) mounted on them, constituting therefore a heterogeneous group. In particular, only one Pioneer has a laser rangefinder system and two Pioneers carry camera systems; but all the Pioneer robots have a gripper and a front ring with 8 sonar discs, except the robot with the laser that also includes a rear sonar ring. All robots of the team maintain the same logical design based on a microcontroller which transfers the readings of the standard elements of the robot (ultrasound sensors, wheel encoders) via RS232, by cable or radio, to a computer client, which can be onboard, where the applications are executed. Finally, the robots are connected by means of a Wireless Ethernet network, which, through several base stations, can access the local network of the university, and the Internet. The robots can be seen in figure 1.6.

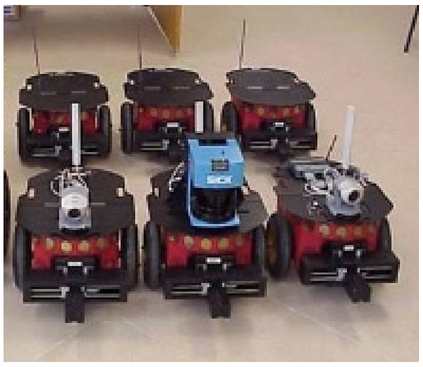


Figure 1.6: The team of Pioneer-2 robots.

- **Mobile Manipulator.** In order to add to the Acromovi architecture the capability to manage new components, a new robot was integrated in the team, a mobile manipulator. The mobile manipulator used in this work is made up of a Mitsubishi PA10 arm with all its accessories and an ActivMedia Powerbot mobile robot, as in figure 1.7. The PA10 arm has 7 degrees of freedom, and is equipped with a two-finger gripper, a force sensor, some infrared sensors for collision detection and a stereo-pair mounted at the end-effector, which consists of two digital IEEE1394 color cameras. At the front of the Powerbot mobile robot there are IR sensors for the detection of stairs and holes, and around it there are bumpers and sonars. Also, it has an onboard computer and a wireless card, which enables it to communicate with other robots or Internet. Finally, the controller of the arm and the rack for batteries are located at the back of the mobile manipulator. This rack provides the necessary power for the arm and makes the mobile manipulator autonomous.

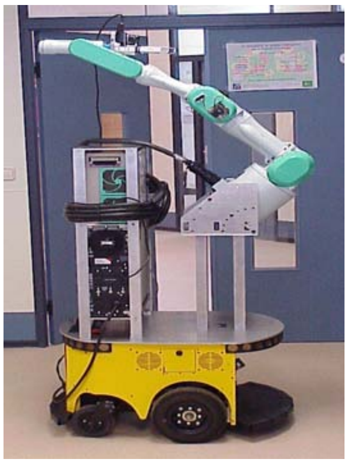


Figure 1.7: Our mobile manipulator (Powerbot + PA10 arm).

- **Linuxbots.** It is desired that Acromovi architecture can control any type of mobile robots. For this reason, a new type of robot was used, the Linuxbots. In the integration of these robots to the Acromovi architecture, concepts of object-oriented programming were used, like superclasses, interfaces or inheritance. The Linuxbots, one of them can be seen in figure 1.8, are a set of mobile robots from the “Bristol Robotics Laboratory” that includes an onboard computer and can communicate with other robots or PC’s by means of a Wireless network. With regard to the sensors, these robots only have one set of infrared sensors facing forward, which allows them to detect obstacles in front of them and no more.

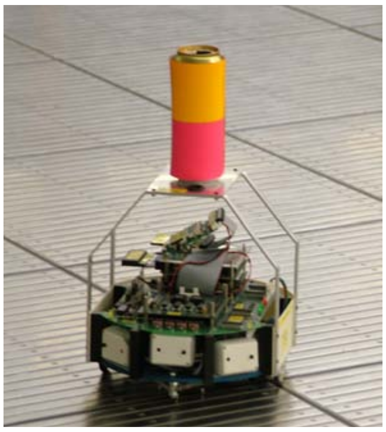


Figure 1.8: One Linuxbot robot.

1.3 Contributions

The main contributions of this thesis are:

- Design and implementation of a control agent-based architecture for the cooperation of a team of heterogeneous mobile robots [263, 265, 266]. This architecture has not only been used for mobile robots, but also to define the cooperation between an arm and a mobile base in a mobile manipulator [269].

- Design of a task of cooperative navigation between several robots by means of the use of the optical flow and time to contact techniques [264, 268]. The most interesting thing is how the robots, one of them with camera and the other without it, cooperate to navigate safely by passing the visual information, represented by the optical flow and time to contact values, from the robot with camera to the others.

- Development of a cooperative task for the creation and navigation of a team of mobile robots maintaining a predefined formation. In this case, one robot carries a laser rangefinder and is the responsible for setting the path to be followed by the formation. Another one includes a camera and is in charge of detecting the rest of the robots in the initialization phase and detecting their real positions while the formation is moving. The rest of the robots do not carry any sensor. Some new methods have been developed:

  - A new method for the visual localization of the robots has been implemented. Using a very common and simple target it is possible to localize one robot and determine its position and orientation with regard to the robot with the camera.
  - An improvement in the calculation of the path to be followed by the formation has been developed. This method tries to soften the sharp angles that are present in the original path calculated by the navigation software of the robots.
  - A new design of formation navigation based on a "follow the conductor" approach, with a series of virtual points with a displacement of the leader for constructing the formation, and Bezier curves for the movement of each robot.
  - Implementation of the cooperation method in the formation task in order to eliminate the odometry errors of the robots and exploit the resources of each robot for the benefit of the whole team. The cooperation is carried out in such a way that the robots without sensors can use the resources of the others to navigate and maintain the formation. For this reason, the robot with the laser sensor indicates the path to be followed and the one with the camera produces information about the errors of odometry to the robots so that they can correct these errors.

- Implementation of a teleoperated system to make the access to all the robots of the team and all their accessories quicker and easier. This system facilitates the learning of the robots and is an important tool for detecting any failure in the development of cooperative tasks.

1.4 Outline of the thesis

The rest of this thesis is organized as follows:

- **Chapter 2** provides a brief description of the approach followed in this thesis and the tools used for it.
- The control architecture implemented for developing cooperative multiagent applications and its evolution are described in **Chapter 3**.
- **Chapter 4** shows how two robots can navigate safely through an environment though only one of them has a camera. For this reason, they have to use the technique of optical flow to detect the information of the environment and cooperate among themselves in order to complete the task successfully.
- How to perform cooperative multirobot formations is shown in **Chapter 5**. The autonomous initialization of the formation and how to construct and maintain the formation is explained. The cooperation is done in such way that one robot with a camera can localize the "blind" robots and help them to maintain the formation.
- Having a simple way to interact with the robots of the team is an important issue. **Chapter 6** provides the description of the system implemented to teleoperate with the heterogeneous team.
- Finally, **Chapter 7** provides some general conclusions and briefly describes some lines of future work.

Chapter 2

Methodology and tools

The state of the art of the techiques used in this thesis is explained in this chapter. Also, the general problem is reduced to the specific sub-problems that have been used in the resolution of the thesis.

2.1 Architectures for robot control

The amount of research in the field of cooperative mobile robotics has grown progressively in recent years [70, 290]. To establish mechanisms of cooperation between robots means that it is necessary to consider the problem of designing cooperative behaviour. That is, having a group of robots, an environment and a task, how the cooperation should be carried out? Such a problem presents several challenges, emphasizing among them the definition of the architecture of the group.

Dean and Wellman [96] state that "an architecture describes a set of architectural components and how they interact". Albus [5] describes an architecture as "a description of how a system is constructed from basic components and how those components fit together to form the whole". Finally, Mataric [233, 238] defines the concept of architecture in the field of robotics as a methodology that "provides a principled way of organizing a control system. However, in addition to providing structure, it imposes constraints on the way the control problem can be solved."

2.1.1 Characteristics of a group architecture

The architecture of a group shapes the infrastructure through which the collective behaviour is implemented. Depending on this architecture, the group will have a different application field and different limitations. The characteristics of the multirobot system will be fixed according to the selection which is made from the various possibilities available. Cao [70] proposed five research axes for classifying a collective of robots: group architecture, resource conflicts, origins of cooperation, learning and geometric problems. Among them, the most important is the "Group Architecture", which is characterised by the following features:

- Centralized vs decentralized.
- Homogeneous robots vs heterogeneous robots.
- Communication structures.
- Modeling of other robots' behaviour.

Centralized vs decentralized

The centralized architectures are those that only have only one control agent. The decentralized ones that don't have an agent of this type, and can be of two types:

- *Distributed*. All the agents have the same control load.

- *Hierarchical*. A control hierarchy exists among the agents.

The decentralized architectures have a major complexity but have some advantages like fault recovery, scalability, reliability and exploitation of parallelism.

Homogeneous robots vs heterogeneous robots

A group of robots is homogeneous if all the robots have been endowed with the same capabilities, and are heterogeneous if they have not.

Heterogeneity adds complexity to the system, task allocation needs to take into account the differences among the robots and try to divide the tasks in an optimal way. Moreover, the robots must be able to model the rest of the robots of the group.

Cao [70] states that the cover of a task is a measure of the grade of cooperation that is needed by a task to be performed. This measure decreases with the growth of heterogeneity. The more homogeneous the group is, the more it is necessary for cooperation to take place in order to perform a task. But homogeneous systems present a major fault tolerance due to the fact that the robots are not specialized.

Communication structures

The structure of communication in a system determines the interaction system of the members. The main modes of interaction are:

- *Environment interaction*. The environment works as communication channel. There is no explicit communication.

- *Sensorized interaction*. The communication occurs through the sensors of each member of the group when it detects the other members. There is no explicit communication. In order to use this type of communication, the members must be able to distinguish the other members of the group from the rest of objects in the environment.

- *Communication interaction*. It uses explicit communication with some type of protocol. Techniques from communications networks are used together with new techniques specially designed for multirobot systems.

Each robot has to decide for itself whether it is necessary to communicate or not in order to perform the tasks, make future plans, obtain information, and so on.

Modeling of other robots' behaviour

The modeling of the intentions, actions, states and capacities of the other robots can be very effective for implementing the cooperation in a system of robots. Whoever or whatever is in charge of modeling has to have some type of predefined representation of the rest of robots. This can reduce the need for communication for the cooperation.

It's important to remark that modeling is of special importance in systems with multiple heterogeneous robots.

2.1.2 Types of architectures

Many different robotic architectures have been defined in the literature. In the following subsections these types are described.

Classical architectures

This approach, also known as deliberative or hierarchical architectures, is the one used from the earliest days of robotics and is based on the ideas of world modeling and planning. In this architecture the control system is broken down into an ordered sequence of *functional components* or *modules*, as can be seen in figure 2.1. This approach is also known as the "sense-plan-act paradigm":

- In the sensing phase, the objective is to get the required information from the available sensors and convert it into some usable form, a model of the world.
- Plan refers to the use of the available information from the sensing phase to determine the control parameters and sequences of actions required for various components in order to make the robot proceed towards the final goal.
- Act phase is the execution of the processes and sequences from the planning phase.

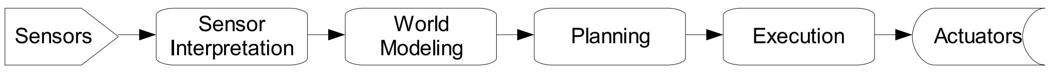


Figure 2.1: Break down of a classical robot control task (adapted from [177]).

The advantages of this approach are the guarantees and optimizations it makes possible. Planning strategies can, in a finite amount of time, compute a sequence of motions that guarantee the successful completion of the task or prove that the task is impossible to carry out. Moreover, a successful plan can be optimized before the robot makes any movement.

But this approach also has some disadvantages, which are increased for mobile robots working in natural or changing environments. These disadvantages are the result of the large amounts of data and the intense computation that are required for modeling the environment or to calculate the plan of actions to perform the task. Also, the time needed to model the environment and to calculate the plan is high and if the environment happens to change during this time, the plan may fail, so it is not able to deal with unknown elements in the scene. In this approach it is very important to have a good balance between time of process and execution.

Another drawback of such architectures is their lack of robustness. Since the information is processed sequentially, a failure in any of the components causes a complete breakdown of the system.

A classical architecture is behind the design of some of the first autonomous robots, such as Shakey [278]. Another examples of this approach are the Handey project [216, 215], CART [259], Nasrem System [218] and ISAM [170].

Behaviour-based architectures

As result of the work of Brooks in the MIT Artificial Intelligence Laboratory [57, 60], an alternative break down of control tasks was developed, giving birth to the “behaviour-based architectures”. Behaviour-based architectures, or reactive architectures, are based on a bottom-up methodology, inspired by biological studies, where a collection of *behaviours* acts in parallel to achieve independent goals. The overall architecture consists of several behaviors reading the sensory information and must be running simultaneously and asynchronously, so a mechanism, a coordinator, must be defined when more than one is in charge of determining the action to be taken by a given actuator, as is shown in figure 2.2.

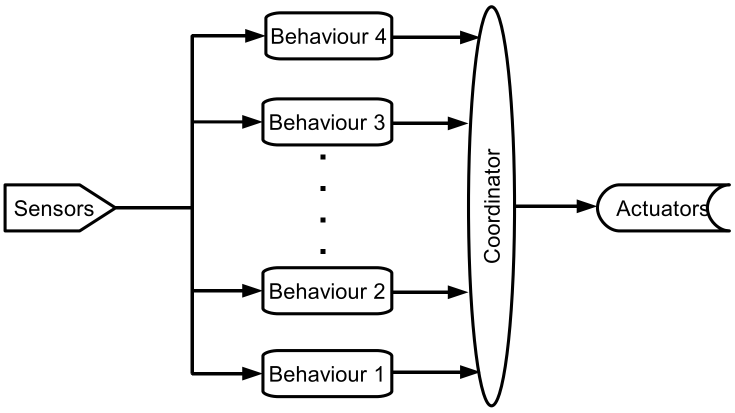


Figure 2.2: Decomposition of a robot control task based on behaviours.

A behaviour is a processing element or procedure and can be considered as a control law in Control Theory [238]. Each behaviour is a simple module that receives inputs from the robot’s sensors, and output actuator commands, as can be seen in figure 2.3. The internal structure of each behaviour can be composed of many modules. However, the behaviours must be independent of each other. These type of architectures are ideal for parallel or decentralized systems, where the behaviours can accomplish possibly competitive goals.

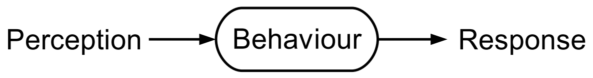


Figure 2.3: Diagram of a single behaviour.

Behaviour-based systems have been proved more appropriate than the classical systems when the real world is not static or cannot be accurately modeled, since they do not need a model of the whole environment [21], only the relevant part of the model is processed in each behaviour. As there is no world model, very little memory is needed in the system. Moreover, due to the fact that behaviours are thought to be simple programs, most of their computations are uncomplicated and can be performed by simple processors within their cycle time. And finally, the most powerful feature is that the intelligence of these systems can be improved incrementally by creating new layers of behaviours. However, not all are advantages, one disadvantage of the behaviour control approach is that the resulting system can't deal with sensor fusion or world modeling. Another one is that it is very difficult to design purely behavioural agents that can learn from experience and improve their performance over time.

Brooks [57] proposed a purely *reactive* behaviour-based system, in which individual behaviours are designed as simply as possible and no internal representation or time history are taken into consideration. Behaviours are grouped into several levels of competence, each level corresponding to a control task. These levels of competence are developed from the lowest to the highest level, following a bottom-up approach, so that each level includes a subset of a lower level. This architecture was named the "subsumption architecture", since each level subsumes the levels over which it is defined. Coordination between levels is achieved by the inhibition or suppression by one module of the output of another module, which provides a replacement output.

Connell extended the work of Brooks introducing a number of new ideas to the subsumption architecture [81, 84, 85] and tested those ideas with "HERBERT", a robot that could collect cans of soda. Moreover, Brooks [60, 59] and Maes [58, 223] perform many experiments with behaviour-based robots, like "GENGHIS", that could walk, climb, collect, wander, hide and learn. Other robots that used this architectures were "TOTO" [235], a robot for hallway-navigating, and "POLLY" [163], a tour-guide robot.

Another well-known behaviour-based architecture, more directly inspired by biological sciences, was introduced by Arkin [17, 18, 21] and is based on results from the *schema theory* [14, 15]. Arkin [21] defined a *schema* as "the basic unit of behaviour from which complex actions can be constructed", and added that "it consists of the knowledge of how to act or perceive as well as the computational process by which it is enacted". In a schema-based architecture, schemes define a collection of behavioural primitives that can act distributedly and concurrently to build a more complex behaviour in response to stimuli from the environment [20, 21]. They can be considered as software objects that can be easily reused [221].

Other behaviour-based architectures have been proposed. Some of them are:

- *Circuit architecture* [180]. Reactive behaviours are expressed using a logical formalism, which allows an evaluation of the performance and properties of the system. Behaviours are grouped into levels of abstraction, and coordination is carried out through arbitration within each level and then between levels.

- *Colony architecture* [82, 83]. Direct descendent of the subsumption architecture, it allows a more flexible specification of the relationships between behaviours. A hierarchical ordering of priorities between behaviours is considered for coordination.

- *Autochtonous behaviours* [152]. The development of this type of architecture has been oriented towards grasping and manipulation tasks, instead of robot navigation. It relies more on models built from sensor data than other behaviour-based systems.

Hybrid and other architectures

It has been proved that behaviour-based architectures are effective in dynamic and complex environments, but they are not always the best choice for some tasks. Sometimes the task to be performed requires the robot to make some deliberation and keep a model of the environment. But behavior-based architectures don't take into account this deliberation and modeling. On the other side, classical or deliberative architectures are not the best choice for tasks in complex environments. Thus, a compromise between these two opposite points of view must be obtained, a *hybrid architecture*.

Hybrid architectures often include at least one part of deliberation, in order to model the world, reason about it and create plans, and another one to represent reactivity, responsible for executing the plans and reacting to any unpredicted situation that may arise. Thus, these architectures present two layers, like in [211]. The architecture described consists of two hierarchical layers for strategic and reactive reasoning. A blackboard database keeps track of the state of the world and a set of activities to perform the navigation. Arbitration between competing activities is accomplished by a set of rules that decide which activity takes control of the robot and resolves conflicts.

But these architectures can also be structured in three layers, as illustrated in figure 2.4: the deliberative layer, with the classical planning to achieve the goals; the control execution layer, which breaks down the plan given by the deliberative layer into smaller subtasks; and the reactive layer, in charge of executing the subtasks set by the control execution layer, implemented with a behavior-based architecture.

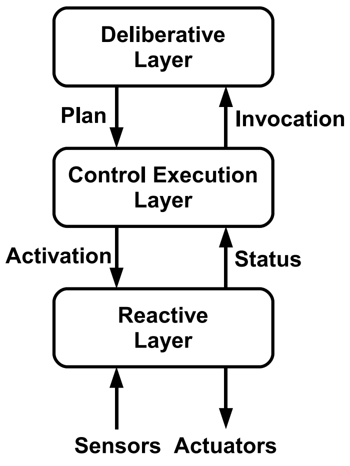


Figure 2.4: Diagram of a three layered hybrid behaviour.

Examples of hybrid architectures are AuRA [16], Atlantis [142], the Generic Robot Architecture [279] or the Animate Agent Architecture [121].

In other approaches, a network of agents have been proposed to control the robot [283].

2.1.3 Multirobot systems

In the study of cooperation among mobile robots, the two most influential disciplines are the Distributed Systems and the Distributed Artificial Intelligence [70].

A system of several cooperative robots is in fact a distributed system, and the most part of the theory of distributed systems can be applied to the study of the cooperation.

The Distributed Artificial Intelligence (DAI) is the subfield of Artificial Intelligence in charge of studying the distributed systems composed by intelligent agents. This field comprises two research areas: Distributed Problem Solving and Multiagent Systems [52].

In Distributed Problem Solving (DPS), the agents solve subparts of the problem in an independent way (task-sharing), and periodically communicate their results among them (result-sharing). The cooperation is divided into three phases: problem breakdown, where the problem is divided in an optimal way and shared among the agents depending on their characteristics; subproblem resolution, each agent solves their part of the global problem; solution synthesis, from the different subsolutions is possible to solve the global problem.

Multiagent Systems (MAS) has centered its research in the collective behaviour of a set of agents with objectives that may be in conflict between them. In that way, the problem is divided and each sub-problem is sent to different problem solving agents with their own interests and goals.

In 1986, Minsky proposed the idea that the brain is composed of independent agents, collectively interacting to produce intelligent behaviour, and proposed the notion of multiagent intelligence systems in which parallel processes interact to produce emergent behaviour [256].

Multiagent systems are an important tool for the development of new robotic applications. They are the natural environment for the development of applications which consist of more than one robot. And the fast implementation of powerful architectures for the specification and execution of tasks for those teams of robots is made possible by their use.

But , what is an agent? Agents have their origin in the Psychology and Distributed Artificial Intelligence, and integrate several aspects as learning, planning, reasoning, knowledge representation. An agent is an entity which can perceive its environment through sensors and act upon that environment through actuators [317]. An agent can be said to be autonomous when its behaviour is based on its own experience, although it can use knowledge previously integrated. In figure 2.5, the relationship between an agent and its environment is shown.

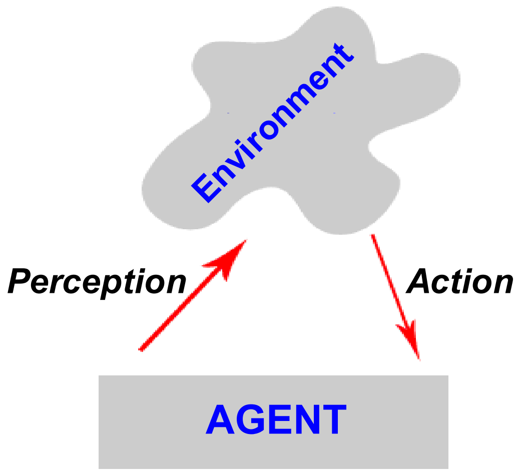


Figure 2.5: Relationship among the environment, the sensors and the agent.

Agents are however seldom stand-alone systems in the environment. In many situations they coexist and interact with other agents in several different ways. A system that consists of a group of agents that can potentially interact with each other is called a Multiagent System (MAS). In order to successfully interact, agents need ability to cooperate, coordinate, and negotiate.

Multiagent systems differ from single-agent systems in that the dynamics of the environment can be varied by the other agents. So, it is possible to think that all multiagent systems have a dynamic environment, where there can be any number of agents with different degree of heterogeneity and with or without the ability to communicate directly among themselves.

Multiagent approaches have already demonstrated a number of important advantages:

- *Co-evolutionary learning*. Many different strategies can evolve simultaneously as they are developed, tested and shared by members of the group.

- *Cooperative behaviour*. Robots can share tasks and information to produce synergistic behaviour. Robots can be specialized to increase efficiency.

- *Distributed processing*. Real-time response can be achieved by distributing the control and information processing among the group.

- *Fault tolerance*. Distributing the task among the team of robots, the group can succeed when one robot might fail.

- *Extended capabilities*. Augment the class of tasks that robots can accomplish.

- *Scalability*. It is easy to add new agents to a multiagent system which can add new capabilities for the whole system.

Multiple agents can improve efficiency in many tasks, due to the fact that some tasks simply cannot be done by a single robot. Multiagent strategies not only increase utility but also allow to develop an important aspect of intelligence: social behavior. Some scientists, such as sociologists and psychologists, use simulated groups of robots to model social behavior in humans.

Multiple scenarios are possible in Multiagent Systems. These scenarios can be organized along two main dimensions: agent heterogeneity and amount of communication among agents [342]. These scenarios can vary from homogeneous non-communicating agents, heterogeneous non-communicating agents, homogeneous communicating agents and heterogeneous communicating agents.

The simplest multiagent scenario involves *homogeneous non-communicating multiagent systems*, all the agents have the same internal structure including goals, knowledge, and possible actions. They also have the same procedure for selecting among the actions. The only difference between the agents are their inputs from sensors and the actions they take at any moment [341]. And of course, they cannot communicate directly between themselves. The communication must be implicit and be done via the environment. A change made in the environment by one agent can be seen by the other agents and then the agent can select the behaviour or action which is stimulated by that change in the environment. In [326], this type of system is used in order to create a robotic football team. Also, in [29], homogeneous, non-communicating agents are used to study formation maintenance in autonomous robots.

In *heterogeneous non-communicating multiagent systems*, the setup gains potential by the addition of heterogeneous agents in the multiagent domain, but the price is a more complex system. In this case, the agents also have different inputs from sensors and take different actions depending on these inputs. But the agents have many more differences, they can have different goals, sets of actions and/or knowledge. In this case, the robots cannot communicate directly among themselves, only by the environment. An example that uses this systems for robotic soccer is explained in [338]. Another example where two robots move continuously to get some food that appears randomly in the environment can be seen in [151]. Finally, in [166] is followed this approach for coordinating the motion of a group of robots.

By using homogeneous agents that can communicate with each other directly, the system changes to a *homogeneous communicating multiagent systems*. With the help of communication, agents can coordinate much more effectively than the other cases. As in the homogeneous, non-communicating case, the agents are identical except that they have different inputs from sensors and take different actions at any moment. But in this scenario, the agents can communicate directly in different ways: by broadcasting or posting on a blackboard for all to interpret, or sending the message point-to-point from one agent to another specific agent.

However, communication also introduces several problems or issues to be solved. One of the most important is exactly what the agents should communicate among them, that is, the content of the communication. Regarding this, agents can communicate several things, among themselves, the states that they have sensed from the environment or they can share information regarding their individual goals. Examples of these systems are [242], where a cooperative distributed monitor is constructed using either stationary cameras and cameras on mobile robots, and [340], where robots communicate internal state information for playing football.

When the system includes agents that may differ in some way, including their data from sensors, their goals, their actions, or their knowledge, then it is an *heterogeneous communicating multiagent system*. In this case, agents can vary in any degree from homogeneity to full heterogeneity. These multiagent systems, which combine heterogeneity and communication, are the ones with more potential, but at the same time are the most complex ones.

In these systems, one of the most studied issue is the communication protocols. In all communicating multiagent systems there must be a common language and a common protocol for the agents to use when interacting. Some aspects to take into account in these protocols are the information in the content of the messages, the format of the message, and the conventions made in order to get coordination.

Some works that implement heterogeneous communicating multiagent systems are: ARCHON [175], one of the first industrial multiagent systems; Creatures [149], that is a a multiagent computer game based on biological models and where agents have the ability to grow and learn based on interaction with human users or other agents; ALLIANCE [287], which use communication among heterogeneous robots to perform tasks divided among the robots; and Millibots [48], that are the smallest components of a heterogeneous multirobot system which is able to perform collaborative communication and mapping. Each millibot is specialized with a different subset of sensors.

Finally, when designing any agent-based system, it is also important to determine how complicated the agents are. *Reactive agents* simply execute behaviours similar to reflexes without maintaining any internal state. On the other hand, *deliberative agents* behave more as if they are thinking, by searching through a space of behaviours, maintaining internal state, and predicting the effects of actions. But, there are also several systems and techniques that mix reactive and deliberative behaviours in an agent.

2.2 Vision for safe navigation

Human and animal vision are the most powerful perception systems. Vision is a sense that consists of the ability to detect the light and interpret it, that is “see”. Vision gets help from multiple information sources to interpret the world. The visual system allows to assimilate information from the environment to help guide the actions.

In the 1980s, researchers such as Arbib [14] began to examine animal models for motor control and perception. They found that a frog uses simple visual motion detection to catch preys and bees center on specific aspects of color spectrum for foraging. They tried to simulate this systems in machines and robots, thus giving birth to the *Computer Vision*. Computer Vision incorporates vision in machines and computers.

2.2.1 Computer Vision

Computer Vision is the branch of Artificial Intelligence that focuses on providing computers with the functions typical of human or animal vision. It is concerned with the theory and technology for building artificial systems that obtain information from images. The image data can take many forms, such as a video sequence, views from multiple cameras, or multi-dimensional data from a medical scanner.

The first use of Computer Vision was made by Larry Roberts in 1961 when he created a program that could “see” a structure of blocks, analyze its content and reproduce it from another perspective. A good text about the essentials of Computer Vision is [161].

Computer vision studies and describes artificial vision systems that are implemented in software and/or hardware. Subdomains of computer vision include scene reconstruction, event detection, tracking, object recognition, learning, indexing, ego-motion and image restoration.

It was not until the late 1970s that a more focused study of the field started when computers could manage the processing of large data sets such as images. However, these studies usually originated from various other fields, and consequently there is no standard formulation of “the computer vision problem”. Instead, there exists an abundance of methods for solving various well-defined computer vision tasks, where the methods are often very task specific and seldom can be generalized over a wide range of applications. In most practical computer vision applications, the computers are pre-programmed to solve a particular task.

A significant part of artificial intelligence deals with autonomous planning or deliberation for system which can perform mechanical actions such as moving a robot through some environment. This type of processing typically needs input data provided by a computer vision system, acting as a vision sensor and providing high-level information about the environment and the robot. As a consequence, computer vision is seen as a part of the Artificial Intelligence field.

Another field which plays an important role in Compute Vision is Neurobiology, specifically the study of the biological vision system. There has been an extensive study of eyes, neurons, and the brain structures devoted to the processing of visual stimuli in both humans and animals.

There are many fields related with Vision. Image Processing and Image Analysis tend to focus on 2D images, how to transform one image to another, e.g., by pixel-wise operations such as contrast enhancement, local operations such as edge extraction or noise removal, or geometrical transformations such as rotating the image.

Computer vision tends to focus on the 3D scene projected onto one or several images, e.g., how to reconstruct structure or other information about the 3D scene from one or several images. Machine

Vision tends to focus on applications, mainly in industry, e.g., vision based autonomous robots and systems for vision based inspection or measurement. This implies that image sensor technologies and control theory are often integrated with the processing of image data to control a robot.

Imaging is primarily focused on the process of producing images, but sometimes also deals with processing and analysis of images. For example, Medical imaging contains lots of work on the analysis of image data in medical applications. Finally, Pattern Recognition is a field which uses various methods to extract information from signals in general, mainly based on statistical approaches. A significant part of this field is devoted to applying these methods to image data.

Examples of applications of Computer Vision systems include:

- *Medical Computer Vision*. It is characterized by the extraction of information from image data for the purpose of making a medical diagnosis of a patient. Image data is in the form of microscopy images, X-ray images, angiography images, ultrasonic images, and tomography images. An example of information which can be extracted from such image data is detection of tumours, arteriosclerosis or other malign changes. It can also be measurements of organ dimensions, blood flow, etc.

- *Industry*. Information is extracted for the purpose of supporting a manufacturing process. One example is quality control where details or final products are being automatically inspected in order to find defects. Another example is measurement of position and orientation of details to be picked up by a robot arm.

- *Military applications*. Examples are detection of enemy soldiers or vehicles and missile guidance.

- *Autonomous vehicles*. Autonomous vehicles use computer vision for navigation, i.e. for knowing where it is, or for producing a map of its environment (SLAM) and for detecting obstacles. It can also be used for detecting certain task specific events, like forest fires. Examples of supporting systems are obstacle warning systems in cars, and systems for autonomous landing of aircraft.

- *Other applications*. Support of visual effects creation for cinema or surveillance.

Each of the application areas described above employs one or several computer vision tasks. Some examples of typical computer vision tasks are presented below.

Recognition

This problem consists of determining whether or not the image data contains some specific object, feature, or activity. The existing methods for dealing with this problem can solve it only for specific objects, such as simple geometric objects (e.g., polyhedrons), human faces, printed or hand-written characters, or vehicles, and in specific situations, typically described in terms of well-defined illumination, background, and pose of the object relative to the camera.

Several tasks based on recognition exist, such as: content-based image retrieval, which consists of finding all images in a larger set of images which have a specific content; pose estimation, that estimates the position or orientation of a specific object relative to the camera; and optical character recognition, which function is to identify characters in images of printed or handwritten text.

Motion

Several tasks relate to motion estimation, in which an image sequence is processed to produce an estimate of the velocity either at each point in the image or in the 3D scene. Examples of such tasks are egomotion, that is determining the 3D motion of the camera, or tracking, that consists of following the movements of objects (e.g. vehicles or humans).

Scene reconstruction

Given two or more images of a scene, or a video, scene reconstruction consists of the computing of a 3D model of the scene. In the simplest case the model can be a set of 3D points. More sophisticated methods produce a complete 3D surface model.

Image restoration

The aim of image restoration is the removal of noise (sensor noise, motion blur, etc.) from images. The simplest possible approach for noise removal is various types of filters such as low-pass filters or median filters. More sophisticated methods imply a model which distinguishes the image from the noise.

Computer vision systems dependant tasks

The computer vision systems depends heavily on the application. Some systems are stand-alone applications which solve a specific problem, and others constitute a sub-system of a bigger application. So, the specific implementation of a computer vision system depends on its functionality.

But there are some typical functions which can be found in many computer vision systems, like image acquisition, consisting of the production of a digital image produced by one or several image sensors; pre-processing, that is any process - resampling, noise reduction, contrast enhancement, scale-space representation - made in the image before applying the image method to assure that the image satisfies certain assumptions; feature extraction, in which image features of interest are extracted from the image data; detection/segmentation, where some points or regions of the image are extracted as relevant for the processing; and high-level processing to execute the remaining image processing from any set of data of the image.

2.2.2 Active Vision

Traditionally, artificial or computer vision has been considered to be “passive”, where the observer is limited to see what it is allowed to see. In this approach, the observer is not capable of choosing how to view the scene, only what is offered, determined by the preset visual parameters and environment conditions [7].

One of the most representative authors that worked with passive vision was Marr, who in the 1980s developed the “computational theory” [230]. He defined visual perception as a reconstruction process of the sensory input,that is a problem which is only solvable under well-defined conditions and assumptions and therefore an ill-posed problem [249]. But this approach attracted a lot of criticism, some of the most important are cited below:

- Excludes the observer’s behaviour, that is, only takes into account the passive vision.

- Does not consider the importance of additional information sources, which are available in biological vision systems, and thinks that vision is enough.
- Considers vision as a general process independent from the observer and the task.
- Tries to collect all the information present in the scene.
- Does not take into account that visual processing is a context-sensitive process.
- Prevents the introduction of learning, adaptation or generalization due to its hierarchical organization of representation scheme.

Active Vision overcomes the mentioned drawbacks of passive vision. Active vision introduces an active observer into the task, that is, the observer performs some kind of activity the purpose of which is to dynamically adjust the parameters (optical or mechanical) of the system in order to simplify the processing for computer vision and improve the results.

An Active Vision system is able to interact with its environment by altering its point of view and by operating on sequences of images rather than on a single frame. As Swain and Stricker [347] characterized, an Active Vision system has mechanisms that can actively control camera parameters such as position, focus, zoom, aperture and vergence (in a two camera system) in response to the requirements of the task and external stimuli.

Active Vision has its origin in the year 1982, when Bajcsy used the term in a NATO workshop. But it was not until the end of the 1980s when basic ideas were suggested by works Bajcsy [26] and Aloimonos [6].

Aloimonos argued that perception is intimately related to the physiology of the perceiver and the tasks it performs. By being active, visual systems in animals control the image acquisition process, thus introducing constraints that greatly facilitate the recovery of information about the 3D scene [7].

After Aloimonos and Bajcsy, many other researchers have extended their work and made important contributions to Active Vision. Some of them are Ballard [32], Eklundh [107], Brown [61], Hynoski and Wu [169] or Swain and Stricker [347] to cite some of them.

But it should be taken into account that active vision also carries a number of subproblems. Some of these problems in which active vision tasks can be involved are: selective attention, foveal sensing, gaze control, data selection from multiple sources, cues from motion, depth cues from binocular disparities or hand-eye coordination among others [108].

Many fields can benefit from Active Vision systems. Some of them include physical aids for visually/physically handicapped, security monitors, unmanned vehicle navigation, crop harvesting, virtual reality, driver assistance, or handwritten recognition [169].

The most obvious advantage of Active Vision is possibly being able to move a camera to enlarge the effective visual field, or to track a moving object while it moves around. However, there are many others like:

- *Depth computation*. Knowing the motion of the camera it is possible to calculate the depth estimation [168, 354] or improve it [24].
- *Self-calibration of cameras*. If the camera can move, it can calibrate itself [37, 55, 314]. The benefit of this technique is that it does not need a predefined pattern for calibration.
- *Contour extraction*. If the observer can move with a motion which is known, then with two successive images it is possible to compute the shape of the 3D contour of an object [6].

- *Active tracking*. It refers to the problem of keeping track of one or several objects over time. The difference from passive tracking is that in this case the camera can be moved to get a higher field of view. One example is explained in [86].
- *Surveillance*. It can take profit from an active point of view in order to classify moving objects and determine what are they doing [128, 281].
- *Motion estimation*. Consists of computing the relative motion of an object or a robot between two consecutive positions. Fixation greatly simplifies the computation by placing the object at the center of the visual field [116].

Active Vision capabilities have had great success in robotics applications, due to the fact that robots offer an excellent way to integrate movement in image processing applications. Although Active Vision can be applied in any field of robotics, the most successful field has been navigation, where Active Vision is used to detect obstacles in the pathway and to look for landmarks for self-localization, thus making navigation safer. Some works that use Active Vision for navigation are described below.

The RHINO project of the University of Bonn investigated autonomous robots [63]. The Rhino robot moved autonomously in office environments applying collision avoidance and path planning. The system used camera, sonar and laser inputs to analyze the current path.

At the UBW of Munich, visual guided vehicles were developed [243]. The vehicles were able to drive autonomously by active control of steering, brakes and accelerator. The entire control system depended on visual input.

In the MOPS project of the institute of robotics at EZH Zurich, autonomous mobile robots as interactive and cooperative machines were investigated [358]. In this project, an autonomous robot was used for internal mail distribution. The robot was able to navigate very accurately based on the visual recognition of natural landmarks.

Another robotics institute, in the Carnegie Mellon University, worked in Automated Highway System (AHS) [40]. The goal of the project was to create the specifications for vehicles and roadways in order to provide hands-off, feet-off, computer-controlled driving.

At the University of Stuttgart, autonomous robots were used for navigation in unknown environments and for recognition of complex and similar objects [282]. At the University of Stanford, autonomous robots were designed to follow objects [42]. Finally, the University of Rochester turned wheelchairs into mobile robots [39] with the aim of following another robot that is manually controlled.

2.2.3 Motion estimation: Optical Flow

As it has been seen, Active Vision systems do not just sense, they act [347]. The most important property of an active Vision system is their interaction with the environment. The systems can change their location, can change the focus, can look to another direction, ... So, the mechanics for moving the camera can not be ignored. One of the most important mechanisms, for moving the camera from one point to another, is the mobile base, which is most usually characterized by a mobile robot.

The use of vision for mobile robot navigation offers some advantages over other types of sensors that make the motion estimation an interesting current field of research. On the one hand, cameras are cheaper than other sensors like laser range-finders, sonar, ... On the other hand, vision data is rich with different types of information about the environment, such as surface color and texture, environment structure, and motion. So, a lot of research has been done in this field.

Since the beginning of time, animals have used motion estimation to survive. Animals based on motion estimation to search for food, to find their mates, to navigate or to detect nearby danger or enemies [195, 260]. If an eye, or a camera, moves in the environment, a flow field appears. This resulting field represents the apparent motion in the image and is called *Optical Flow*.

In fact, if a series of images are taken in time, and there are moving objects in the scene, or it is the camera itself which is moving, useful information about what the image contains can be obtained by analysing and understanding the difference between images caused by the motion. One of the main characteristics that can be extracted from the optical flow is the velocity of an object, and thus, it is possible to have a notion of the position that the object will have in the next instant of time. This is very important in order to prevent possible collisions among objects while navigating.

Optical flow can be defined as the distribution of the apparent velocities of movement of the brightness pattern in an image, and arises from the relative movement of some objects and a viewer [162]. Also, Barron [35] defines the optical flow as the computing of the approximation to the 2-D motion field from a projection of the 3-D velocities of surface points onto the imaging surface from spatio-temporal patterns. So, the optical flow describes the direction and the speed of the motion of the features in the image.

Supposing that the illumination of the environment is constant, at least during a short period of time, a pixel maintains its intensity when it moves. Extending this fact to all the pixels of the image, it is possible to deduce an equation that indicates velocity.

Being $I(x, y, t)$ the intensity of the pixel (x, y) of the image in the instant t. Then,

$$I(x + \delta x, y + \delta y, t + \delta t) \approx I(x, y, t) \tag{2.1}$$

where $(\delta x, \delta y)$ is the displacement of the image after time δt. Expanding the left-hand side of the equation in a Taylor series,

$$I(x, y, t) + \delta x \frac{\delta I}{\delta x} + \delta y \frac{\delta I}{\delta y} + \delta t \frac{\delta I}{\delta t} + O^2 = I(x, y, t) \tag{2.2}$$

where O^2 contains the second or higher order terms, which are assumed insignificant. Substracting $I(x, y, t)$ on both sides, dividing by δt, and ignoring O^2,

$$\frac{\delta I}{\delta x}\frac{\delta x}{\delta t} + \frac{\delta I}{\delta y}\frac{\delta y}{\delta t} + \frac{\delta I}{\delta t} = 0 \tag{2.3}$$

Using the abbreviations

$$u = \frac{\delta x}{\delta t}, v = \frac{\delta y}{\delta t}, I_x = \frac{\delta I}{\delta x}, I_y = \frac{\delta I}{\delta y}, I_t = \frac{\delta I}{\delta t}$$

where (u, v) are the x and y components of the optical flow vector, it is obtained the following equation,

$$I_x \cdot u + I_y \cdot v + I_t = 0 \tag{2.4}$$

This image is known as the *optical flow constraint equation*, and serves as basis for all the techniques that calculate optical flow.

But optical flow calculation has several problems, and it is generally an ill-posed problem itself [116]. Insufficient texture or illumination makes the determination of the optical flow a difficult problem. Also, due to the aperture problem, only the normal optical flow can be estimated, that is, the flow in the normal direction of the spatial gradient [117, 302].

Also, optical flow calculations are usually very computationally expensive. An additional drawback derives from the fact that after the calculation of the optical flow, another calculus is needed to interpret the information, so, the time of computation is increased. Finally, other disadvantages are caused by possible occlusions or on the transparency of any obstacle.

In order to get the best results for the calculation of the optical flow, the environment must satisfy a series of requisites:

- There must be uniform illumination
- The object must not reflect in an anomalous way
- The object must move in a parallel plane to the image plane

Almost 30 years ago Horn & Schunck [162] published the seminal paper on optical flow calculation. Since, then, many techniques have been proposed to compute the optical flow. Some surveys can be seen in [41, 140]. Barron in [35] classifies optical flow algorithms by their signal-extraction stage. This provides four groups: differential techniques, energy-based methods, phase-based techniques and region-based matching.

Differential Techniques

Differential techniques are characterized by a gradient search performed on extracted first and second order spatial derivatives, and temporal derivatives. From the Taylor expansion of the flow constraint equation, the gradient constraint equation is obtained:

$$\nabla I(x, y, t)(v_x, v_y, 0) + I_t(x, y, t) = 0 \tag{2.5}$$

Horn and Schunk [162] combine a global smoothness term with the gradient constraint equation to obtain a functional for estimating optical flow. Their choice of the smoothness term minimizes the absolute gradient of the velocity. This functional can be reduced to a pair of recursive equations that must be solved iteratively.

Lucas and Kanade [217] also construct a flow estimation technique based on first order derivatives of the image sequence. In this case, a pre-smoothing of the the data before using the gradient constraint equation has been chosen.

Fleet and Langley [123] try a more efficient implementation using IIR temporal pre-filtering and temporal recursive estimation for regularization. The temporal support was reduced to 3 frames, and the computation time improved, while only slightly diminishing performance.

Spetsakis [334] uses an adaptive Gaussian filter approach to minimize the loss of occluding information. Spetsaskis derives a measure of confidence called the incompatible measure. If the residual is high, the solution is considered robust. If the residual is low, then the Gaussian has not gathered enough information and its size must be increased.

Energy-Based Methods

The advantage of energy-based methods is the hierarchical decomposition of the image sequence in the frequency domain. Energy based techniques extract velocities by using families of band-pass filters that are velocity and orientation adjusted. The Fourier transform of a translating 2-D pattern is

$$\hat{I}(\vec{k}, w) = \hat{I}_0(\vec{k})\delta(w + \vec{v}^T\vec{k}) \tag{2.6}$$

where $\hat{I}_0(\vec{k})$ is the Fourier transform of $I(\vec{x}, 0)$, $\delta(k)$ is a Dirac delta function, w denotes temporal frequency, and $\vec{k} = (k_x, k_y)$ denotes spatial frequency. This implies that all power associated with the translation will be mapped to a plane that traverses the origin in frequency space.

Heeger [157] uses a family of twelve Gabor filters of different spatial resolutions to extract velocity information from the image sequences. By using Gabor filters [122], which provide the simultaneous spatio-temporal and frequential localization, a clean band-pass representation is obtained. A least square fit is then applied to the resulting distribution in frequency-space.

Phase-Based Techniques

Phase-based methods use a family of velocity-tuned filters to extract a local-frequency representation of the image sequence. Flow estimates are provided by gradient search in the phase space of the extracted signatures. The most accurate optical flow estimations are produced by the phase-based approach in [35, 212].

Motivation for this approach is based on the argument that evolution of phase contours provides a good approximation to projected motion field. The phase output of band-pass filters is generally more stable than the amplitude when small translations in the scene are sought.

Region Matching

Region matching forms the filters for feature extraction from the previous image in the sequence. Tiles from the previous image are correlated with the next image using some distance measure. The best match provides the most likely displacement.

The distance measure used by more classical algorithms such as Anandan [9], and Singh and Allen's [331] is referred to as the sum-of-square differences (SSD).

Anandan [9] constructs a multi-scale method based on the Burt Laplacian pyramid [67]. A coarse-to-fine strategy is adopted so that larger displacements are first determined from less resolved versions of the images and then improved with more accurate higher resolution versions of the image. This strategy is well suited for large-scale displacements but is less successful for sub-pixel velocities.

Singh and Allen [331] provide another approach using the SSD. They use a three-frame approach to the region matching method to average out temporal error in the SSD. For a frame 0, they form an SSD distribution with respect to frame -1 and frame +1. From that, a probability distribution is built up. Then, a Laplacian pyramid is employed. This provides a more symmetric distribution about displacement estimates in the SSD.

Benoits and Ferrie [47] build a more robust and simplified region matching metric that is similar to the SSD. Thresholding is applied to the difference and sum distributions. When the sum of pixel intensities is small, the data is considered unusable. When the difference between successive inputs is small, the intensities should be considered the same. From the total difference-sum ratio distribution, average pixel-matching error is summarized.

Another interesting region matching flow algorithm is Camus' quantized flow [69]. Camus constructs a real-time flow algorithm with the idea that performing a search over time instead of over space is linear in nature rather than quadratic. A quantized sub-pixel displacement field results. The success of this algorithm is based on the idea that support for a faster frame-rate reduces the necessary area of spatial search by providing a better sampling rate.

Other Algorithms

In [213], using a general steady 3-D motion model in combination with 3-D Hermite polynomial differentiation filters an efficient and accurate algorithm is constructed. Their approach requires generating a family of spatio-temporal filters. The filters are thus tuned to reflect the 3-D motion model when projected onto a 2-D perspective model.

Other original approaches to flow estimation include Nesi [274] work. They obtain discontinuous flow estimates using clustering techniques. They use the Combinatorial Hough Transform to propose a multipoint solution with most-likelihood estimation. They argue that clustering techniques provide a better approach to solving the flow-blurring problem than more traditional techniques that use least-square estimation.

Heitz and Bouthemy [158] use a statistical model to provide estimation of discontinuous flow fields. They suggest that using the intersection of solution sets provided by complementary constraints provides a more robust estimate. Data fusion between constraints is formulated using a Bayesian framework associated with Markov Random Fields.

Finally, among the applications of the optical flow the following studies should be mentioned: approximations to image motion [3, 115, 144, 254, 271, 375], motion segmentation [49, 53, 106, 109, 172, 261, 313], computing of the focus of expansion and time to collision [66, 171, 284, 304, 343, 345], motion compensated image encoding [73, 104, 374], computing of stereo disparity [34, 89, 173, 197], measuring of blood flow and heart-wall motion in medical imagery [301], and measuring of minute amounts of growth in corn seedlings [36, 210].

2.2.4 Time to Contact

Most of the existing work in optical flow-based navigation has concentrated on the development of low-level behaviours for specific navigation tasks. These behaviours can be classified into one of four categories of operating scenarios which are considered to encompass the major conditions under which optical flow has been used for navigation [244]. These categories are:

- *Near-constant, time-critical conditions*.The optical flow induced by robot motion remains constant during navigation but regular control updates are required to maintain stability. This relates to environmental conditions which are relatively free of obstacles, but may contain boundaries (such as walls or the ground surface). This encompasses such behaviours as corridor centring or flight stabilisation.

- *Near-constant, non-time-critical conditions*. The environments described are as above, but the optical flow estimation is not time-critical with respect to the needs of control. This is most used in tasks such as visual odometry.

- *Continuous, rapidly changing conditions*. A robot is making continuous motion approaching a surface or surfaces. This can be associated with obstacle avoidance, docking, or graze landings in flight. In these cases, the optical flow exhibits rapidly increasing flow. This effect is used to provide a measure of range, based on a time-to-contact estimate.

- *Non-continuous, rapidly changing conditions*. The frame-to-frame changes in flow are greater than the temporal development of flow. This is concerned with robots capable of 3D motion (aerial or underwater) performing visual behaviours such as hovering or station keeping.

In this case, vision for safety navigation, the behaviour that better fulfills the necessities of the task is the “continuous, rapidly changing conditions”, and the work is centered on the calculation of the *time-to-contact* in order to avoid the obstacles that can appear during the navigation task.

The time-to-contact can be computed from the optical flow extracted from monocular image sequences acquired during motion. It is defined as the time needed for the robot to reach to the object or obstacle, if the instantaneous relative velocity along the optical axis is kept unchanged [250].

Many different techniques has been proposed in the literature since Lee presented his seminal work in 1976 [200, 201]. Also, Nelson and Aloimonos, in [273], wrote one of the first papers where vision-based spatio-temporal techniques enabled a robot to avoid collisions by means of the computing of the time-to-contact.

Finally, using the optical flow field computed from two consecutive images, it is also possible to find the depth information for each flow vector by combining the time-to-contact computation and the robot’s speed at the time the images are taken. Simply, with the product of the velocity of the robot and the time-to-contact value it is possible to know the depth for each pixel of the image [333].

2.3 Cooperative formations of multiple mobile robots

As it was seen in chapter 1, in recent years there have been several advances in robotics that have made it possible to work with teams of multiple mobile robots, and consequently, to research into how these teams can cooperate in order to accomplish tasks in a coordinated way.

A considerable amount of research has been focused on the problem of coordinated motion and cooperative control of multiple autonomous robots in recent years. Researchers have been trying to understand how a group of moving robots can perform collective tasks.

Research on coordinated robots started after the introduction of the behaviour-based control paradigm [57]. Before this, research had been concentrated on controlling single robot systems and distributed problem-solving systems which does not involve robotic components.

In these early days of the distributed robotics, the topics of main interest included:

- *Cellular (or reconfigurable) robot systems*. The main examples of this topic are works on the cellular robotic systems [137], and swarms [46].
- *Multirobot motion planning*, with works on traffic control [300], and movement in formations [12, 361].
- *Architectures for multirobot cooperation*, such as the work on ACTRESS architecture [22].

After that, several researchers began investigating issues connected to multiple mobile robot systems. Since then, this field has grown with a great variety of topics, but due to its youth, no topic area can be considered to be mature enough [292]. Some of these areas have been explored more extensively and the researchers are beginning to understand how to develop and control certain aspects of multirobot teams. These areas or topics are described below.

- *Biological systems*. This area is centered in examining the social behaviours of insects and animals, and to apply this research to the design of multirobot systems. Common applications are flock, disperse, aggregate, forage, or following trails [98, 103, 234], where the rules of biological societies such as ants, bees or birds are imitated in the control of multirobot systems. Moreover, it has been demonstrated how cooperation can emerge as result of acting on selfish interests [246]. Cooperation in more complicated animals, such as wolfs, has also been studied and it has generated advances in cooperative control, in particular in predator-prey systems, although only in simulations [45, 156]. A domain that uses competition in multirobot systems is multirobot soccer [231, 339]. Finally, more recent topics include the use of imitation for learning new behaviours, and physical interconnectivity, used by insects, to allow collective navigation over challenging terrains.

- *Communication*. This topic involves the study of the effect of the communication on the performance in multirobot teams in different tasks. The communication can provide benefits for particular types of tasks [222]. Also, in many cases, the communication can produce a great benefit [27]. Recent work on this topic focuses on representations of languages and the grounding of these representations in the world [167, 179].

- *Architectures, Task planning, and Control*. In this case, the focus of the research is the development of architectures, task-planning capabilities, and control in order to focus on issues such as action selection, delegation of authority and control, communication structure, heterogeneity versus homogeneity in robots, achieving coherence in local actions or resolution of conflicts. In general, the architectures developed tend to focus on providing a specific type of capability, such as task planning [4], fault tolerance [289], swarm control [237] or human design of mission plans [248].

- *Localization, Mapping, and Exploration*. A lot of research has been carried out in this area, but only in single robot systems. Only recently, these areas have been applied to multirobot systems, but trying to extend the single-robot developments to multiple robots instead of developing a new distributed algorithm. Some subcategories in this field are the use of landmarks [97], scan-matching [64], or graphs [305] for the localization, mapping or exploration. Also, it is possible to use range sensors (sonar or laser) or vision sensors.

- *Object transport and Manipulation*. In order to carry, push, or manipulate common objects, a cooperative approach is desired. Many projects have taken this into account dealing with a lot of variations of this task including the number of robots or the type of multirobot systems. The most important subfield is box-pushing, where a lot of research has been done [191, 240, 289, 315, 336]. This is due to the fact that this task is easier than the carry task, where several robots must grip a common object and navigate to a destination in a coordinated way [159, 184, 363].

- *Reconfigurable robotics*. This topic involves all those robotics systems where robots are capable of reconfigurating themselves, that is, allowing individual modules, or robots, to connect and reconnect in various ways to generate a desired shape in order to perform a desired function. With these systems, there is an increment of robustness, versatility and self-repair. One of the most popular applications is navigation, where these systems show different configurations, such as rolling track motion [371], earthworm or snake motion [74, 371], and spider or hexapod motion [74, 371]. Also, other systems are able to connect in different shapes forming matrices or lattices for specific functions [51, 316, 356, 373]. Research has only recently begun in this field, and most of the systems developed are not yet able to achieve anything except simple laboratory experiments or in simulations.

- *Learning*. Typical tasks of this area include predator/prey [45, 156], box-pushing [224], foraging [237], multirobot soccer [308, 339], and cooperative target observation [291]. In particular, it is desired that these tasks are cooperative, that is, they cannot be decomposed into independent subtasks to be solved by a distributed robot team. By means of cooperation, the system manages to learn.

- *Motion coordination.* This is one of the most popular fields of study in multirobot systems. Some applications that are well understood include multirobot path planning [370], traffic control [300], formation generation [12], and formation keeping [29, 361], however, not many demonstrations of these applications use physical robots, only simulators. Other fields of study related with motion coordination are target tracking [291], target search [198], and multirobot docking behaviors [257].

To sum up, there are some of these areas that are well understood, such as biological systems, the use of communication in multirobot teams, and the design of architectures for multirobot control. In others, the progress has been considerable, such as multirobot localization/mapping/exploration, cooperative object transport, and motion coordination. And others are now getting good results, such as reconfigurable robotics and multirobot learning.

2.3.1 Motion coordination: multirobot formations

In the field of motion coordination, one of the tasks where most success has been achieved is in the organisation of formations of multiple robots. Having a group of robots moving in formation would be beneficial in many applications in real world environments, such as exploring an area for surveillance, search and rescue, demining or military missions, surrounding and capturing a prey, convoying, grazing, or cleaning tasks, to cite some of them.

Formations can be defined as groups of mobile robots establishing and maintaining some predetermined geometrical shape by controlling the positions and orientations of each individual robot relative to the group, while allowing the group to move as a whole.

Formation behaviours in biological systems, like flocking or schooling, benefit the animals that use them. Animals in a group also combine their sensors to maximize the chance of detecting predators or to more efficiently forage for food. Since groups of artificial robots could similarly benefit from formation tactics, robotics researchers have drawn from these biological studies to develop formation behaviors for both simulated agents and robots.

An early application of artificial formation behavior was the behavioral simulation of flocks of birds and schools of fish for computer graphics. Important results in this area were originated in a pioneering work [307]. But more realistic simulated fish schooling by accurately modeling the animals' muscle and behavioral systems was developed in [355]. Also in [54], a system was developed for realistically animating herds of one-legged agents using dynamic models of robot motion.

Another topic that has produced attention is the dynamics and stability of multirobot formations. In [362], a strategy was developed for robot formations where individual robots are given specific positions to maintain relative to a leader or neighbour. But it did not include integrative strategies for obstacle avoidance and navigation. Moreover, in [77], formation generation by distributed control is demonstrated, where large groups of robots are shown to cooperatively move in various geometric formations. In this case, the research is also centered on the analysis of group dynamics and stability, and not in obstacle avoidance.

Mataric has also been investigating emergent group behavior [234, 236]. It is demonstrated that simple behaviors like avoidance, aggregation and dispersion can be combined to create an emergent flocking behavior in groups of wheeled robots.

Other papers related to formation control for robot teams include [138, 285, 369, 372]. In [285], Parker studies the coordination of multiple heterogeneous robots. Yoshida [372], and Yamaguchi [369], investigated how robots can only use local communication to generate a global grouping behavior. Similarly, Gage [138] examined how robots can use local sensing to achieve group objectives like coverage and formation maintenance.

Finally, Parker, in [286], simulates robots in a line-abreast formation navigating past waypoints to a final destination. This approach includes a provision for obstacle avoidance. Parker's results suggest that performance is improved when agents combine local control with information about the leader's path and the team's goal.

In this type of task, it is necessary to solve different problems in different fields of application. In the local field, the problem is to determine what each robot has to do to maintain the formation, and in the global field, what the group must do in order to accomplish its objectives.

Different characteristics can be associated with these coordination strategies. In [251], these characteristics, grouped in three different categories, are outlined.

- **Perceptual characteristics**
  - *Visibility of other robots*. The visibility of the robots in the group can be complete [324] or limited [119, 132].
  - *Frame of reference*. It is possible that robots use an absolute reference system to decide their decisions [203, 324], or do it in relation to their own reference system [119, 132].
  - *Communication capabilities*. Three options are possible: no communication [324], communication of global information [119, 132, 203] or communication of local information [132, 203].
- **Formation characteristics**
  - *Types of formation*. Robots can be organised to move into many different types of formation, such as circle, diamond, wedge, line, column, triangle, rectangle, arrowhead, hexagon, tree, lattice (hexagonal, rectangular or triangular), or arbitrary.
  - *Position determination*. This refres to how the robots calculate their position in the formation. Three techniques exist: "unit-center-referenced", where the point used as reference for all the robots is calculated as the average of the x and y positions of all robots [29]; "point-referenced", where a single point, which can be the leader position [29] or a virtual point [199, 203], is used as a reference for each robot to determine its position; and finally, "neighbour-referenced", in which each robot determines its position in relation to one [132] or two [119] of its neighbours.
  - *Structural constraints*. Formations can be rigid [203], so they preserve the shape of the formation at all time, or flexible [119], when the formation can change.
- **Control characteristics**
  - *Decision process*. Although almost all the systems are distributed, it is also possible to use a centralized approach [199]. In the case of distributed control systems, two different types of decision process are available, homogeneous, where all the robots take the same decision rules, or heterogeneous, if the robots have different rules about taking decisions.

- *Dependence on temporal states*. The robots can determine their decisions only from the information that they collect from their sensors at that instant or determine their decisions based on information that has been acquired in the past [132, 346].
- *Control strategy*. Some strategies can be taken into account, based on control laws like input-output feedback linearization [199], behavior-based [29, 132] or hybrid [120], which can involve a supervisory level and an execution level.

But for the creation of formations by a robot team, many different types of coordinations problems must be solved [31, 131, 133, 187, 202]. These problems are enumerated below:

1. How can formation shape be maintained while moving [29, 119, 120, 199, 203, 324]? In this case, criteria like the maintenance of stability while moving in formation and a robust response to robot failure are important [132].

2. How should obstacles be avoided? Two options are possible when the formation finds an obstacle in its path, one possibility is to occasionally split or deform and then reestablish the formation [29, 119, 120, 324], or the other possibility is to preserve the shape of the formation [132] and to move the entire formation in a way that makes avoiding the obstacle possible.

3. How should the shape of the formation be changed while it is moving [99, 132, 324]?

4. How should the formation be initialized and established [324, 346]? This point refers to the need to specify if the formation is assembled at the starting point and/or established. That is, do the robots organize themselves in the formation or is it necessary for a human agent to place the robots in their initial position?

5. Determination of feasible formations [120, 348]. That is, is it possible to make any type of formation, or is the number of formations limited due to restrictions in the system?

2.4 Teleoperation systems for mobile robots

From the very beginning, men have used tools to increase their manipulation ability. At the beginning, it was only woods used to make fall the mature fruit of the trees. Some types of tools, such as the blacksmith tongs, have been used to transport or manipulate red-hot or materials potentially dangerous, such as radioactive materials. The development of tools for operating at a distance culminated in the development of teleoperation systems. Teleoperation is a form of control in which a human operates a machine at a distance. When this machine is a robot, it is called telerobotics.

Since the first developments for teleoperation, the nuclear industry has been the main consumer of teleoperated systems. But nowadays, teleoperation has several applications in other sectors. Specialists use teleoperated systems to explore hazardous environments. Military personnel increasingly employ reconnaissance drones and telerobotic missiles. At home, we have remote controls for the garage door, the car alarm, and the television. A brief summary of different applications is below detailed.

- **Space applications**. These applications include planetary experimentation and exploration, and satellite maintenance and operation.

- **Nuclear applications**. In this case, the aim is to manipulate and experiment safely with radioactive substances, to operate and maintain installations, such as jets, pipes, installations for nuclear fuel production, ..., pollution-free installations, or intervention in nuclear disasters.

- **Underwater applications**. Some of the applications are: inspection, maintenance and construction of underwater installations, underwater mining, or inspection of underground water.

- **Military applications**. There have been many significant developments in this field because most of the mobile teleoperation technologies have been developed for military applications. In this case, examples go from remote monitoring systems to the use of UAV (Unmanned Air Vehicles). Also, ground vehicles, called UGV (Unmanned Ground Vehicle), are used to inspect, explore and or rescue. And USV's (Unmanned Surface Vehicle), have been developed for force protection, surveillance, mining detection, and special forces operations.

- **Medical applications**. These applications range from prosthesis or devices for assistance for handicapped people, to telesurgery or telediagnosis. The most important development in this area is the master-slave systems for performing "robot-attended surgery", in which a surgeon uses mechanical arms that reproduce the movements that he performs in the patient. The whole operation is supervised by the surgeon by means of cameras.

- **Other applications**. Other developments include building and mining, maintenance of voltage lines, intervention in natural disasters, car industry, or entertainment.

Teleoperation means simply operating a device at a distance, so that it enables the capability to sense and to act in a remote location. A basic teleoperation framework is composed of:

- Human operator, who performs the control of the operation at a distance, generating commands via input/control devices and receiving feedback from displays. His control can be continuous or intermittent in order to monitor and indicate objectives and plans.

- Remote device, which can be a manipulator, mobile robot, vehicle or similar device. It is the device working in the remote location and executes the operator's commands. Usually, it is equipped with sensors, actuators and often some level of autonomy.

- Communications channel. It is the set of devices which modulate, transmit and adapt the signals that are transmitted between the local and remote locations. Usually, wireless connections or Internet are used for this purpose.

Another important element in any teleoperated system is the interface. The interface is the group of devices which allow the interaction of the operator with the teleoperation system. Usually, it is formed by all the control devices, as joysticks, mouse, keyboard, haptic devices, ... and a set of displays with feedback information from the remote site, as video, sonar readings, laser readings, force feedback, ... In figure 2.6, it can be seen the relation among all the systems involved in the teleoperation.

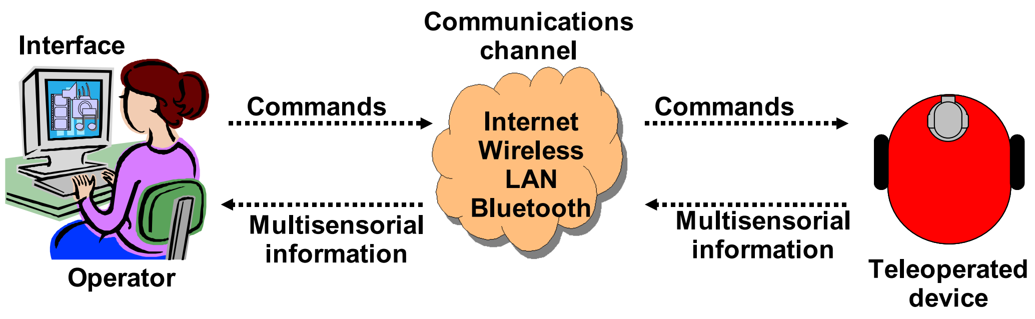


Figure 2.6: General diagram of a teleoperation system

2.4.1 Control models for teleoperated systems

Depending on the autonomy level of the robot and the role the the human plays in the decision making, there are three basic control models used to operate with robots: direct control, supervisory control, and fully autonomous control.

The *direct control* model has been in use since the early 1900's. *Supervisory control* was developed in the 1960's as a way to characterize operators functioning in discontinuous control loops [327]. *Fully autonomous control* describes robot systems that rely on humans to set high-level goals or to specify high-level strategy, but which then execute independently of direct human control.

Direct Control

This is the most common method for performing robot teleoperation. In it, the operator directly operates the remote robot using hand-controllers (e.g., joystick, mouse, keyboard) while monitoring video on one or more displays, and the robot must reproduce the movements as exactly as possible. Although the robot or the user interface can assist the human, the primary responsibility for perceiving the remote environment, making decisions, and executing the plan rests with him.

Direct control is the cheapest and easiest model to implement. However, it can be problematic due to the fact that all control decisions depend on the human, so, tasks can only be performed while the human remains in the system. Moreover, performance is limited by the human capabilities and workload. Many factors can influence the perception of the environment by the human, such as sensorimotor limits, knowledge, skill, training, control station design or communication bandwidth. All these factors can lead to the human operator making mistakes. Another limitation is that this model demands expert users because of the difficulty and the risk is high.

Supervisory Control

The supervisory control concept appeared from research on earth-based teleoperation of lunar vehicles [118], and the term supervisory control is derived from the analogy between a supervisor's interaction with subordinate staff [327]. Although it is used primarily for telemanipulation [50] in known and structured environments, it has been also used for mobile robots teleoperation [2, 25, 208, 330, 337, 364].

In the supervisory control, an operator divides a problem into a sequence of sub-tasks which the robot can perform on its own. Once control is given to the robot, it is expected to complete the tasks without human intervention. During autonomous action, the operator watches displays, which show sensor data processed by the robot perception modules, to monitor the progress and to understand how the robot is performing the task. If a problem occurs, the operator is responsible for finding a solution and for developing a new task plan. So, all the control is made through a user interface

But Sheridan [327] notes that the human is not restricted to only to a supervisory role. Instead, he can intermittently control the robot, or he can control some variables while leaving the others to the robot. The former approach is known as “traded control” and the latter as “shared control”.

The main problem of this model is that failures can occur if the human operator is unable to recognize that the robot is ill-suited for the situation. Another limitation is that it is unable to accommodate a wide range of users, although it has less impact than in the direct model due to the fact that the robot is able to execute certain tasks autonomously.

Another problem relies on the human’s ability to understand the remote environment and process. The human must maintain situational awareness, must know what the robot is doing, and must recognize when the robot has problems. If the user interface is poor, or if the human builds an incorrect mental model of what is happening, it can result in damage or loss to the system.

Fully Autonomous Control

In fully autonomous control, the human gives high-level goals or tasks to the robot, which autonomously achieves them. The robot is able to plan prior to execution, concurrently with execution, or continuously during execution. This approach is what research in “autonomous mobile robots” wants to achieve.

Fully autonomous control can be seen as supervisory control taken to its extreme, where the human gives a high-level, abstract goals which the robot then achieves by itself. Also in this model, the human interacts with the robot through the user interface. While the robot operates, the primary function of the human is to monitor execution via interface displays.

In this case, as the model is a simplification of the supervisory model, the same limitations are applied to it, but in this case, they less depends on the capabilities of the human operator.

Other control models

Other control models have been recently implemented in teleoperation systems. These new models take into account the relation between the operator and the remote device. That is, they not only assume that the remote device is a tool, but also an active element in the system.

The first of these types is called *collaborative control*, which is a particular instantiation of supervisory control [127, 126]. In this type of control, the remote device and the operator dialogue to decide which action must be carried out. This relation between the operator and and the device improves the operating capacity of the system. However, one problem is that it prevents the operator from acting when the robot is not capable of continuing its plan.

Other example is *cooperation control* [147, 349]. In this case, the remote device and the operator perform cooperated tasks each one adopting different roles depending on the task. The operator can adopt different roles ranging from coordinator, when it only supervises the operation of the remote device, to device partner, where operator and device work independently to achieve common objectives.

2.4.2 Interface models for teleoperated systems

The basic function of a teleoperation interface is to enable an operator to remotely perform tasks. Thus, all interfaces provide tools and displays to perceive the remote environment, to make decisions, and to generate commands. A properly designed interface enables an operator to be more productive.

The most difficult aspect in robot teleoperation is that the operator cannot directly perceive the remote environment. Instead, he relies on the interface to provide him with sensory information. As a result, the operator often fails to understand the remote environment and makes errors. Some common problems are: poor performance, poor judgement, loss of situational awareness, and failure to detect or to avoid obstacles. A poorly designed or implemented interface aggravates these problems. Thus, in order to achieve a successful robot teleoperation, it is necessary to build effective and easy-to-use interfaces.

Controlling mobile robots through teleoperation is a challenging task that demands a flexible and efficient user interface. Mobile robots are often equipped with numerous sensors (proximity sensors, system status sensors, positioning and heading devices, multiple cameras, ...) that provide a large quantity of data to the user. Because the amount of data is too large to fit on the screen, and because the data can change rapidly and unpredictably depending on events in the environment, teleoperation interfaces often display user-selected data with a windowing paradigm that facilitates quick display modification.

Although in conventional teleoperation, the user interface serves only the operator with displays that provide information for human decision making, most modern user interfaces are designed with user-centered approach. In user-centered design, the basic goal is to facilitate the activity of humans in the interface, for that, they enable humans to do things faster, with fewer errors, and with greater quality [275].

Four main classes of robot teleoperation interfaces are most commonly used: direct, multimodal/-multisensor, supervisory control, and novel. *Direct interfaces* contain all "traditional" vehicle tele-operation systems based on hand-controllers and video feedback. *Multimodal/multisensor interfaces* provide multiple control modes or use integrated display techniques such as augmented reality or virtual reality. *Supervisory control interfaces* are designed for high-level command generation, mon-itoring and diagnosis. Finally, *novel interfaces* use unconventional input methods (e.g., gesture) or are intended for unusual applications. But there are also some interfaces, particularly hybrid systems, which are difficult to put into any of these categories because they combine elements from more than one class.

Direct Interfaces

The most common method for mobile robot teleoperation is using direct interfaces, where the operator directly operates the remote robot using hand-controllers, such as joysticks, while it is monitoring video images from cameras mounted on the robot. This type of interface is appropriate when real-time human decision making or control is required, and the environment can support high-bandwidth, low-delay communications. If not, direct control can be tedious, fatiguing, and error prone [327].

But robot teleoperation using direct interfaces can be problematic. Loss of situational awareness, inaccurate attitude and depth judgement, failure to detect obstacles, and poor performance are com-mon occurrences [247]. Other research has tackled sensorimotor requirements for teleoperation [189] and the impact of a video system field of view on remote driving [146].

Direct interfaces are used for applications requiring navigation through unknown environment (survey, inspection, reconnaissance, ...) and robot positioning in support of remote tasks (loading, construction, ...). Other applications are target detection and identification [299], tunnel and sewer reconnaissance [194], remote mine survey and rescue [154], telepresence controls for submersibles [33] or heavy work vehicles [155].

Multimodal and Multisensor Interfaces

When the robot operates in changing, complex, and highly dynamic environments, it can be difficult for the operator to perceive the remote environment, and consequently it is very difficult to make control decisions in time. Multimodal and multisensor interfaces can help the operator by providing a rich information feedback and command generation tools. These interfaces provide the operator with several control modes and information displays.

Multisensor displays combine the information from multiple sensors or data sources from the robot or the environment to present a single integrated view. These displays are very important for those tasks in which the operator must rapidly process large amounts of multiple and dynamically changing data.

Multisensor displays have been used in military aircraft to improve cockpit efficiency by fusing information from imaging sensors and databases [351]. Also, they have been used to operate vehicles in hazardous environments and for tasks requiring multiple control modes [125, 196, 296]. Moreover, multisensor displays can improve situational awareness, facilitate depth and attitude judgement, and speed decision making [102, 277].

Supervisory Control Interfaces

In this type of control, as explained before, the operator divides a problem into a sequence of subtasks, and the robot then executes this sequence on its own, the operator playing the role of supervisor of the work done by the robot. So, supervisory control interfaces are designed for high-level command generation, monitoring and diagnosis.

These interfaces, in contrast with direct control interfaces, are well suited for applications which are limited to operate with low-bandwidth communication links or in the presence of high delay, due to the fact that the remote robot includes some level of autonomy. Thus, the work of the operator is focused on navigation and motion command generation. So, supervisory control interfaces must provide tools to make these tasks easier.

Some of these facilities are designed for task planning and sequence generation. Moreover, some interfaces provide methods for reviewing and analyzing results, so that the operator can monitor task performance and identify anomalies. Additionally, interfaces can also make use of haptic devices with force feedback or virtual reality devices, all these focused on giving to the operator the maximum grade of telepresence in the remote environment.

But there are still many design challenges for these interfaces including display layout, human-robot interaction, and adaptability (to the situation, to the user, ...). Also it is important to provide in the interfaces a way for exchanging information between the operator and the robot, above all in those situations when the robot is not able to perform a task and the operator needs to know what has happened. Another important design issues is that the interfaces must provide flexible displays.

Supervisory control interfaces have been developed for numerous ground vehicles, where the most popular command mechanism is waypoint driving. In this type of driving, the operator specifies a series of intermediate points which must be passed on route to a target position. Waypoint driving can be performed using either maps, which requires accurate localization and maps, or images, which is more suitable for unexplored environments. Some examples are explained in [88, 182, 241, 303, 365], most of them based on interplanetary teleoperation.

Novel Interfaces

The last category of vehicle teleoperation interfaces are the novel interfaces. Interfaces can be novel in several ways. Some interfaces are novel because they use unconventional input methods, such as a hands-free remote driving interface based on brainwave and muscle movement monitoring [8], or haptic interface which enables an operator to "drive-by-feel" and vision-based system which maps hand movements to motion commands [124].

Another way to create novel interfaces depends on the control device used for allocating the teleoperation interface, as can be the use of PDA's [124, 296]. Using PDA's it is possible to create lightweight, portable, and feature touch-sensitive displays interfaces in portable devices.

Moreover, personal interfaces can also be considered novel since they began to use the Internet for teleoperating robots in 1994. Since then, several web-based robot teleoperation interfaces have been developed [150, 253, 328, 329].

Finally, interfaces can be novel if they are used in unusual ways, such as in [294], where the teleoperation interface enables operators to "project their presence into a real remote space", that is, the teleoperated robot can be seen as "real-world avatar" for the operator.

2.5 Proposed approach

In this section, the different techniques and options chosen to implement the points of this thesis are described.

Architecture for robot control

As it has been said before, one of the objectives of this thesis is the development of an architecture for a team of heterogeneous mobile robots so that they can cooperate.

Following the premises seen in subsection 2.1.1 for defining the characteristics in a group architecture, it is possible to define the use of a decentralized distributed architecture for multiple heterogeneous robots, which can use communication interaction and in which the modeling of the rest of robots is made by the programmer.

Regarding this type of the architecture, as seen in subsection 2.1.2, the architecture implemented in this thesis is a layered hybrid architecture. This architecture is divided into two layers grouped in a middleware layer, the lower for reactive behaviours such as the execution of actions in any actuator or getting information from any sensor, and the upper for deliberative tasks, such as the localization or the navigation of the robot. Upon this middleware layer is the applications layer, that is also modeled as a deliberative layer.

Finally, the multiagent systems implemented is of the class "heterogeneous communicating multiagent systems", whose characteristics can be seen in subsection 2.1.3. Moreover, the type of agents used in the architecture can be seen as reactive or deliberative, depending on which layer they are situated in.

Vision for safe navigation

In order to validate the architecture created, a cooperative application is implemented which consists of making possible for a group of robots to cooperate in order to safely navigate through an unknown environment.

As seen in previous sections, to navigate safely it is necessary the use of vision, or more specifically active vision, to estimate the motion of both, the obstacles that can appear in the way of the robot, and the motion of the robot itself. In this estimation, one of the techniques most uses is the calculation of the optical flow field.

In the case of this thesis, the optical flow field is calculated to manage the robot while navigating. Among all the possible techniques available in the literature to calculate the optical flow, a differential technique has been chosen, and more specifically, the one based on the Horn and Schunk calculations.

Also, in an attempt to detect the obstacles on the path of the robot and to know the time remaining for colliding with them, the time-to-contact is calculated. Moreover, from this time-to-contact, it is possible to calculate the distance to the object, a calculus that is also used in the development of the application.

Multirobot formations

Using the different categories shown in section 2.3.1, it is possible to outline the characteristics of the system used in this thesis for the multirobot formations.

In this case, the robots only have limited visibility, they only know their actual position and only one robot is equipped with a camera to recognize if the rest of the robots are maintaining the formation or not, and only one robot is equipped with a laser range-finder in order to allow it to follow a predefined path on a predefined map.

Also, they use a relative frame of reference, that is, they do not use an absolute positioning system, only the robot with the laser uses an absolute system in relation with the map that it uses to navigate. And in relation with the communication capabilities, the robots can share local and global information to compensate for their limited visibility, in fact, the global information shared among the robots is the actual position of the robot with the laser, and the local information is the actual position of each robot determined by the robot with the camera.

Due to the fact that the robots must be localized by the robot with the camera in both the initialization of the formation and during the maintenance of the formation, all the possible formations that can be performed must avoid the occlusions of the robots from the point of view of the robot with the camera. In chapter 5, some diagrams are shown as examples.

Each robot determines its position in order to maintain the formation with respect to the leader or a displacement of the position of the leader, with a “virtual” point. And although in the actual implementation of the formations is rigid, it is possible to make it flexible in an easy way.

Regarding the control characteristics, a distributed approach is used in order that each robot decides what to do in each moment from the information that it has at that moment. But the global decision process can be seen as centralized, because the leader is which decides how to move the formation. No information from the past is used in the application. And finally, the control strategy used in this case is hybrid strategy, because an execution level is used in each robot to react to the movement of the formation, and a supervisory level is used in order to supervise that each robot is in the correct position.

Moreover, almost all the coordination problems associated with the problem of the multirobot formation have been taken into account, and the ones that have not been taken into account have an easy solution with the system implemented. In this case, the application has been centered on the problems of autonomous initialization and maintenance of the formation shape. Also, the determination of the possible formations has been addressed. The other two problems, avoidance of obstacles and the change of the shape of the formation, have not been taken into account, but an easy solution is possible in the framework defined for the implementation of the applications of multirobot formations.

Teleoperation of a team of mobile robots

The main objective of the teleoperated system developed is to allow any user, independently if he knows the system or the instructions to operate a robot, can teleoperate with the team of robots to achieve a specific task goal, making transparent for that user the differences among the different robots of the team.

The aim is to allow the user or operator to give orders to the robots to move them in the desired direction with the desired velocity, to be able to pick up an object or move the camera to focus toward a desired direction. At the same time, the aim is to allow the user to see by means of the interface the consequences of the actions executed and all the available information relative to the robots.

To get that, two main decisions have been taken in order to implement the control and interface models of the system. In order to implement the control model, a supervisory traded control model has been chosen. This model has the advantages of direct control when the aim is to operate with the robots and from the supervisory control when the aim is only to monitor the activity of the team of robots. Moreover, it allows the robots to be operated in intermittent way.

To design and implement the interface model, the multimodal/multisensor model has been chosen. This interface means that all the information about the robot can be integrated into an unique display. As the robots have different types of information sources, this is the best option for implementing the interface. Furthermore, due to the differences among the robots of the team, a different display is available for each robot. These displays are generated in execution time depending on the elements that each robot has active at that moment.

Chapter 3

Acromovi architecture

This chapter describes the architecture designed in order to solve the problems of cooperation.

3.1 General overview

The architecture forms the backbone of a robotic system. The right choice of the architecture can facilitate the specification, implementation and validation of the applications implemented for the system. This chapter presents an architecture for the development of collaborative applications for a heterogeneous team of mobile robots.

Acromovi architecture was born from the idea that teamworking is an essential capability for a group of multiple mobile robots [178, 239]. As it has been said previously, having one single robot with multiple capabilities may waste resources. Different robots, each one with its own configuration, are more flexible, robust and cost-effective. Moreover, the tasks to achieve may be too complex for one single robot, whereas they can be done effectively by multiple robots [13].

In the decade of 90's, there was an enormous increase in the amount of research on robot groups focused on architectural issues for robot control and coordination [19, 29, 136, 190, 232], but heterogeneous groups were not studied in so much detail. Heterogeneous robot systems are now a growing focus of robotics research. A team of robots is heterogeneous if the members of the group are different in at least one of the following attributes: mechanics, sensing, computing hardware, or nature of on-board computation [344]. These teams presents some advantages, for example some members of the team are strong, others fast, others small and agile, others powerful enough for complex computation, ... Thus, each robot can include certain special features which equip it to perform the tasks assigned to it. However, groups of heterogeneous robots are more complex and costly to mantain and design.

The main idea in heterogeneous robot teams is that each robot can also use the equipment in other robots in order to compensate for the functions they do not have. In this way, the architecture can implement applications for a team of heterogeneous robots that can help each other by sharing capabilities [219, 220].

The Acromovi architecture defines a framework for application development by means of embedding agents and interfacing agent code with native low-level code. This architecture emphasizes the reusability of software, allowing the programmer to seamlessly integrate native software components (vision libraries, navigation and localization modules), by means of agent wrappers, providing the programmer with a set of tools for mobile robots. And, what is more important, it allows the robots to share resources among themselves and obtain easy access to their elements by the applications. Finally, other important characteristics of the Acromovi architecture are the scalability and ease-of-use.

Because the architecture is just a set of embedded agents, a multiagent system programming tool has been selected for its implementation. The chosen multiagent tool was JADE (Java Agent DEvelopment Framework). It is a software framework for developing agent-based applications in compliance with the FIPA specifications for interoperable intelligent multiagent systems [43]. But other implementations have been tested, as will be seen in section 3.3, based on other techniques. The first one is based not on agents, but on a set of software components implemented by Java RMI [255]. The second one is also based on multiagent systems, but in this case the tool chosen was MadKit [114]. A brief performance test is shown to demonstrate that the implementation with JADE is the most suitable for the implementation of the architecture.

3.1.1 Related work

The amount of research in the field of cooperative mobile robotics has grown progressively in recent years [70, 290]. Some examples of these works, centered on the some of the characteristics of Acromovi architecture, are presented below.

In an attempt to use traditional AI techniques, the *GOPHER* architecture was developed to resolve problems in a distributed manner by multirobots in internal environments [68]. A central tasks planning system (CTPS) communicates with all the robots and has a global vision of the tasks that have been done and of the availability of the robots to perform the tasks that remain to be done.

An example of a heterogeneous behaviour-based architecture is *ALLIANCE*, which is designed to organise the cooperation of a small-medium team of heterogeneous robots, with little communication among them [289]. It assumes that the robots are relatively able to discern the effects of their actions and those of the rest of the agents, either through its perception or through communication. Individual robots act based on behaviours or sets of behaviours to carry out their tasks.

A learning variant of this architecture, *L-ALLIANCE* [288], uses communication among heterogeneous robots to help divide tasks among the robots which they can carry out independently. Agents only broadcast the task on which they are currently working. If the communication fails, multiple robots might temporarily try to do the same task, but they will eventually realize the conflict by observation and one will move on to a different task. In L-ALLIANCE, robots learn to evaluate each other's abilities with respect to specific tasks in order to more efficiently divide their tasks among the team.

ABBA (Architecture for Behaviour Based Agents) [178] is another behaviour-based architecture for mobile robots, the purpose of which is to design an architecture to model the robot's behaviour so that it can select reactive behaviours of low level for its survival, to plan high level objectives oriented to sequences of behaviours, to carry out spatial and topological navigation, and to plan cooperative behaviours with other agents.

Another interesting project, *MICRobES*, is an experiment of collective robotics that tries to study the long time adaptation of a micro-society of autonomous robots in an environment with humans. Robots must survive in this environments as well as cohabit with the people [298].

CEBOT (CEllular roBOTics System) is a hierarchical decentralized architecture inspired by the cellular organization of biological entities. It is capable of dynamically reconfigurating itself to adapt to environmental variations [134]. It is composed of "cells", in the hierarchy there are "master cells" that coordinate subtasks and communicate among themselves.

SWARM is a distributed system made up of a great number of autonomous robots, usually homogeneous. The term “SWARM intelligence” has been coined to define the property of systems of multiple non-intelligent robots to exhibit a collective intelligent behavior. Generally, it is a homogeneous architecture, where the interaction is limited to nearest neighbors [176].

The University of Ulm has created a middleware for autonomous mobile robot applications. *Miro* is based on the construction and use of object-oriented robot middleware to make the development of mobile robot applications easier and faster, and to foster the portability and maintainability of robot software [357]. This middleware also provides generic abstract services such as the localization or behavior engines, which can be applied to different robot platforms with virtually no modifications.

In an attempt to use agents, dynamic physical agents are considered in an architecture called DPA, being well suited for robotics [280]. A physical agent is the result of integrating a software agent in an autonomous hardware. This hardware is frequently a mobile robot. *DPA architecture* shows the agent as a modular entity with three levels of abstraction: control module, supervisor module and agent module. The architecture’s operation is based on two basic processes for each module: the process of introspection and the one of control, and in a protocol of intermodular dialogue which allows the exchange of information between modules.

In order to reuse native software components, agents can be embedded in an interface level or middleware. *ThinkingCap-II* [76] is an architecture developed, in a project of distributed platforms based on agents for mobile robots. It includes intelligent hybrid agents, a planning system based on visual tracking, vision components integration, and various navigation techniques. Furthermore, it has been developed over a real-time virtual machine (RT-Java), implementing a set of reactive behaviors.

Another system that uses native software in order to encapsulate drivers and algorithms for mobile robots is *CARMEN* (Carnegie Mellon Robot Navigation Toolkit) [258]. CARMEN is an open-source collection of software for mobile robot control that provides modular software for mobile robot control and navigation including: base and sensor control, obstacle avoidance, localization, path planning, people-tracking, and mapping. But it is oriented for using in a unique robot and moreover, the communication between CARMEN modules is handled using a separate package called IPC. So, it is not suitable for the implementation of cooperative tasks.

Finally, one of the most widely used software nowadays for programming multirobot applications is the *Player/Stage* project [143]. Player is a network oriented device server that provides clients with network-oriented programming interfaces to access actuators and sensors of a robot. It employs a one-to-many client/server style architecture where one server serves all the clients of a robot’s devices and it relies on a TCP-based protocol to handle communication between client and server. Accompanying Player is the robot simulator Stage, a lightweight, highly configurable robot simulator that supports large populations of robots and allows programmers to control these virtual robots navigating inside a virtual environment.

These works implement architectures and systems so that teams of mobile robots can carry out cooperative tasks. The architecture implemented in this thesis goes some steps further emphasizing the reusability of software and allowing the programmer to seamlessly integrate native software components. It is also emphasized the sharing of resources from a robot among the whole team and the easy access to the physical elements of the robot by the applications. And adding to all these the versatility and power of the multiagent systems for the resolution of cooperative tasks for a group of heterogeneous robots.

3.2 Design of the architecture

This section develops the conceptual design of the architecture. This architecture, as it has been said in the previous section, consists of a set of embedded agents that can cooperate among themselves in order to perform complex tasks.

Multiagent systems are an important tool for the development of new robotic applications. They are the natural environment for the development of applications which consist of more than one robot and make possible the fast implementation of powerful architectures for the specification and execution of tasks for those teams of robots.

The embedded agents that constitute the Acromovi architecture work as interfaces between the applications and the physical elements of the robots. Some agents also make the handling of these elements easier and provide higher-level services. The embedding agents are the key for powerful distributed applications being rapidly developed.

Furthermore, multiagent systems are composed of multiple interacting agents. These agents communicate in order to achieve more efficiently their goals or the goals of the society in which they exist. To make possible these communication acts it is necessary to define protocols, sets of rules which guide the interaction among the agents. Interaction enables agents to cooperate and to coordinate in achieving their tasks.

3.2.1 Overall description

The robots that constitute the team, the Pioneer-2 robots, already have a two-layer programming architecture with native (C/C++) libraries for all their accessories. The lower layer called ARIA [311] and the upper layer called Saphira [186]. However, this native code is oriented to both local and remote processing from only one controller, without defining collaboration mechanisms between robots or applications.

Acromovi architecture tries to overcome this problem. Acromovi is based on a hybrid layered architecture as presented in 2.1.2. In this architecture each one of the layers can interact with those above and below it, and the agents can interact with another agent through direct communication at each of the layers.

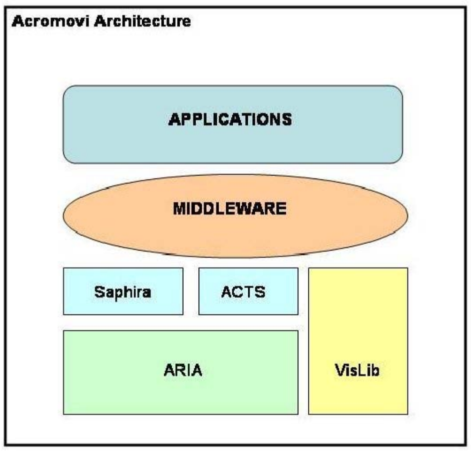


Figure 3.1: General architecture diagram

As can be seen in figure 3.1, Acromovi architecture is a distributed architecture that works as a middleware of another global architecture for programming robots [263]. It is situated between the native robot architecture and the applications and it allows the collaboration and cooperation of the robots of the team. This middleware is based on an agent oriented focus.

Also, Acromovi architecture is able to subsume any other extra software, like the ACTS [309] (a color tracking library) and the Vislib [312] (a framegrabbing library). Subsuming existing C/C++ libraries is a powerful and quick method for rapidly developing Java code and thus embedding agents with all the functionality of native libraries.

As shown in figure 3.2, the middleware level has been divided into two layers. As in any hybrid architecture, the lower layer involves processes for sensing and perception and for motion control, which are directly connected to the sensors and actuators of the robot. The upper layer usually implements processes for world modeling and for planning and deliberation.

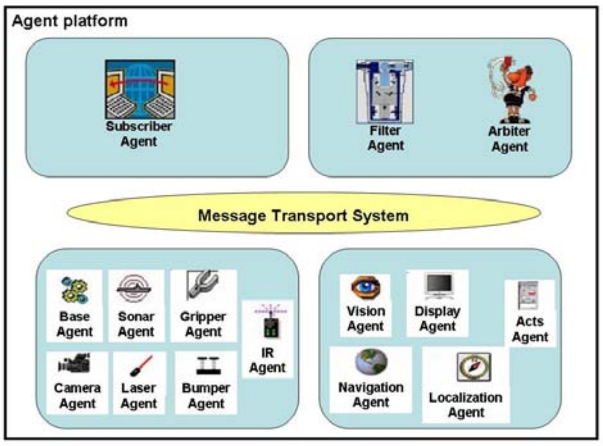
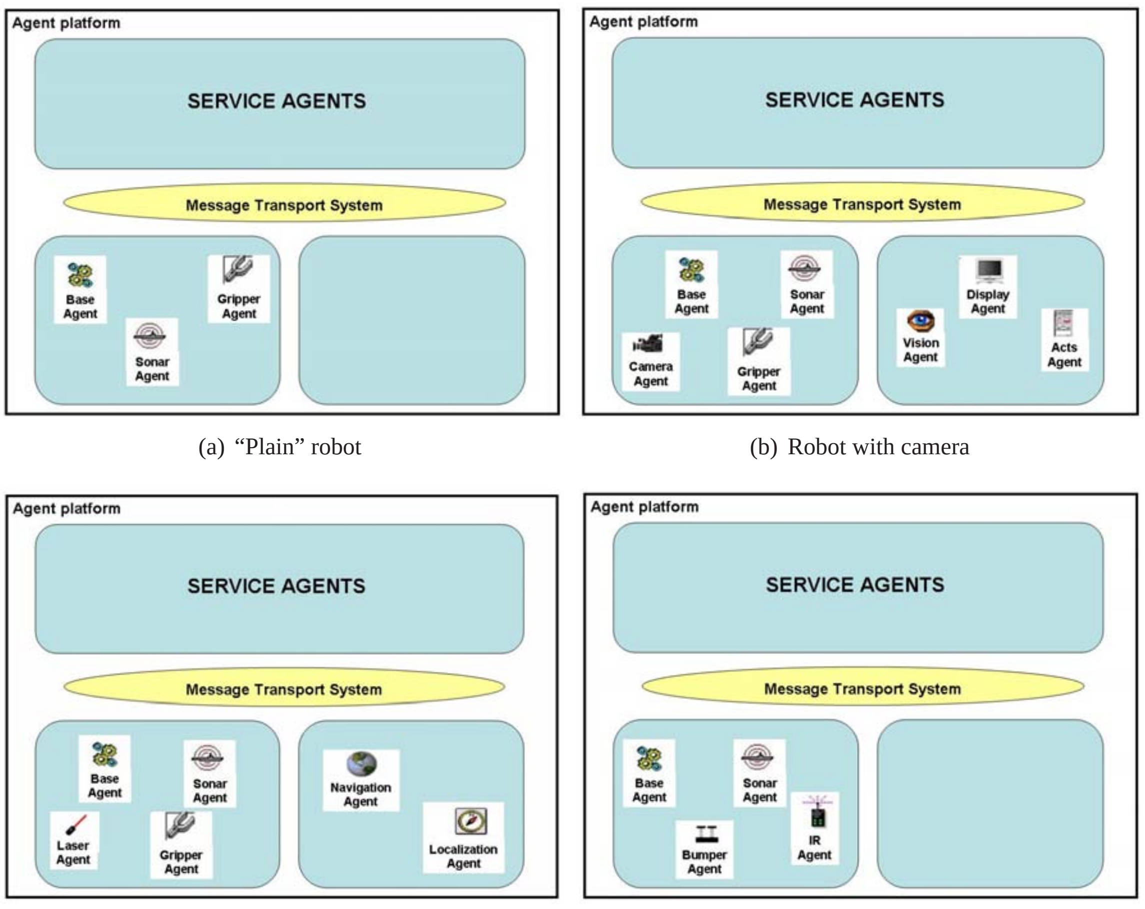


Figure 3.2: The middleware layer of the Acromovi architecture

In the lower layer, there is a set of components which are in charge of handling the physical elements of the robot, such as the sonar, laser, base, ... The others are special components that offer services to the upper layer, such as vision, navigation or localization. These components only perform two kinds of tasks or functions, the processing of the requests that receive and the sending of the results, in a reactive operations mode. Thus, they could be implemented as software components, but for reasons of efficiency, implementation facility and integration with the whole structure, they have been developed as agents that encapsulate the required functionality.

The upper layer of the middleware comprises a great variety of embedded and supervising agents, which need more deliberate processing. These agents act as links between the applications and the agents that access the physical components of the robot, e.g. agents for monitoring the state of the agents/components of the lower layer, agents that provide services of subscription to a certain component, agents that arbitrate or block the access to a component, and any other type of agent which might offer a service that can be of interest for the whole architecture or for a particular application.

Because of the heterogeneous character of the team and depending on each robot in particular, there are different middleware layers for the robots. These middleware layers are generated in execution time according to the elements that each robot has active at the moment of its execution. These different configurations can be seen in figure 3.3.



(a) “Plain” robot (b) Robot with camera

(c) Robot with laser (d) Powerbot

Figure 3.3: Different middleware layers

Finally, above the middleware layer is the applications layer, that is also implemented as agents. These applications, to access to the physical components of the robots, must communicate with the agents of the middleware, which then access the native layer that physically controls the robot.

An important feature of the created architecture is its scalability. That is, the architecture can grow in a fast and easy way. Once an application has been tested, and if it is useful for the entire architecture, it can be easily converted into a new agent of the upper layer of the middleware. From that moment, this new agent is in charge of offering new services, that can be helpful, to another applications that could be created. Thus, each application that we make can expand our system to make applications more complex and more interesting, following a bottom-up design.

3.2.2 Agents description

As has been said, the elements of the robot are represented and managed by agents. The agents that form the lower layer have been conceptually divided into three different groups, depending on which part of the robot the agent carries out its work with. Thus, there are *body agents* (base, gripper, sonar, bumper and IR agents), *laser agents* (laser, localization and navigation agents) and *vision agents* (camera, ACTS, vision and display agents).

The body agents are the basic agents for the operation of the robot and they correspond with the main physical elements of the robot.

- **BaseAgent** is in charge of everything that is related to the movement of the motors and to the internal state of the robot.
- **GripperAgent** is in charge of the operation of the gripper, and of allowing the other agents to manage it.
- **SonarAgent** is in charge of the correct operation of the sonar, and of returning the appropriate values when these are required by other agents or applications.
- **BumperAgent** manages the bumpers of the robot and returns their values when they are required.
- **IRAgent** is devoted to the handling of the 4 infrared sensors equipped on one of the robots. The agent gives information about the state of these sensors when it is necessary.

The laser agents communicate with the laser device and allow the use and management of the localization and navigation modules.

- **LaserAgent** allows multiple operations with the laser and is in charge of its correct operation.
- **LocalizationAgent** is responsible for localizing the robot in a known environment.
- **NavigationAgent** calculates the best path to follow between two points and moves the robot along this path.

The vision agents are in charge of capturing the images from the camera and process them in a correct way.

- **CameraAgent** physically moves the camera and gives information about its state.
- **ACTSAgent** is used for the color tracking of objects.
- **VisionAgent** captures images, and can visualize or modify them by means of the Vislib library.
- **DisplayAgent** modifies and processes operations on the captured images, by means of the use of the Java 2D libraries.

The agents of the upper layer have a generic character and can take different forms depending on the agent of the lower layer that they are serving. Thus, there might be more than one agent of the same type but serving different agents of the lower layer. These agents have also been divided into two groups. On the one hand, the generic **SubscriberAgent** is in charge of controlling those agents of the lower layer that can serve information or data in a continuous way. On the other hand, the **FilterAgent** and **ArbiterAgent** are in charge of managing the operation of those agents of the lower layer that may have conflicts in its access, such as concurrency problems. The agents of the lower layer are those ones that can physically move any part of the robot, such as the base, gripper or camera.

3.2.3 Interaction protocols

As in any other multiagent system, Acromovi is composed of multiple interacting agents. These agents need a way to communicate between themselves in order to achieve their goals or those of the society in which they interact. The tools that make this communication possible among the different agents of a system are the protocols.

In the domain of multiagent systems, a protocol is a set of rules which guides the interaction that takes place between several agents and that governs how the information is delivered. These rules define what messages are possible for any particular interaction state. That interaction has to be carried out so that agents exchange information. Interaction enables agents to cooperate and to coordinate in achieving their tasks.

There are two different types of protocols to be defined in a multiagent system, the communication and the interaction protocols. *Communication protocols* enable agents to exchange and understand messages. *Interaction protocols* are defined as a series of communication acts, forming conversations, aimed at achieving some form of coordination among the agents..

As Acromovi architecture has been implemented by means of the JADE [43], the communication protocols used are those offered by this framework, such as TCP/IP, HTTP, RMI, ... The advantage offered by JADE is that it permits a transparent use of this protocols, adapting the transport mechanism to each situation. Moreover, it incorporates a transport mechanism that works like a chameleon because it adapts to each situation in the best way, by transparently choosing the best available protocol.

On the other hand, the interaction protocols used are the ones proposed by FIPA [332]. FIPA proposes a set of interaction protocols for multiagent systems. JADE defines a library with all these interaction protocols ready to use. But these protocols are very general and they don't deal with any of the features of the Acromovi architecture. So, two new protocols that enable the system to carry out new functions have been implemented.

The first protocol implies a modification of the FIPA subscription protocol, in which an agent requests regular notification of the required information. In this new protocol two types of agents are involved: the *subscribers* that want to have a regular flow of information from one of the elements of the robot, such as the readings of the sonar and laser or a continuous flow of images from the vision system; and the *providers* that serve these flows of information from the agents that manage the elements of the robot. These agents are closely joined to the agents that manage the elements of the robot. That is, there is a provider joined to the sonar agent, another joined to the laser agent, and so on.

The provider agents are continuously requesting information from the agents that manage the elements of the robot, so they always have the actual values of these elements. The rate at which these provider agents request the information from the other agents is given by the real rate at which the physical elements can serve the information. The operation of the protocol can be seen in figure 3.4.

The subscriber agents in the moment of the subscription indicate how often they want to receive the data from the provider agent ("SUBSCRIBE X"). The provider agent serves the data to each subscriber agent at the right moment ("DATA n") depending on the frequency indicated in the subscription message. That is, if one subscriber wants to get the data just one for every two times that the sonar gets data, the provider agent sends to the subscriber agent the data every two readings of the sonar. To do that, the provider agent has an active list which points out the agent which has made a petition of subscription and the frequency with which this agent wants to get the data.

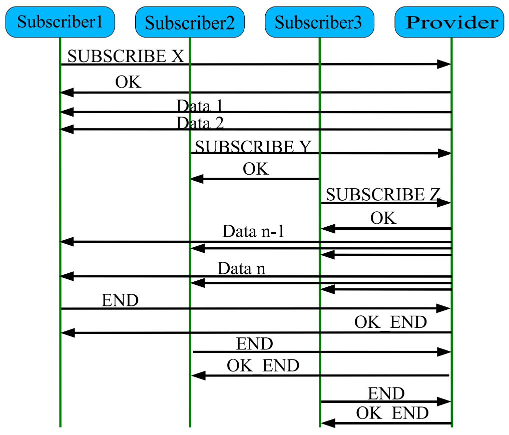


Figure 3.4: Subscription protocol

The second protocol is a new one and has been introduced in order to control the access to some "problematic" elements, which are all those elements that go wrong if one or more agents can access and execute operations at the same time on them, eg. the base, the gripper, the camera ... So, this protocol tries to resolve the concurrency problems. Another important characteristic of this protocol is that it prevents the agents from carrying out operations with the "problematic" elements that can put the robot in danger, eg. if an agent wants to move the robot two meters, but there is a wall only one meter away. The conceptual operation of the protocol can be seen in figure 3.5.

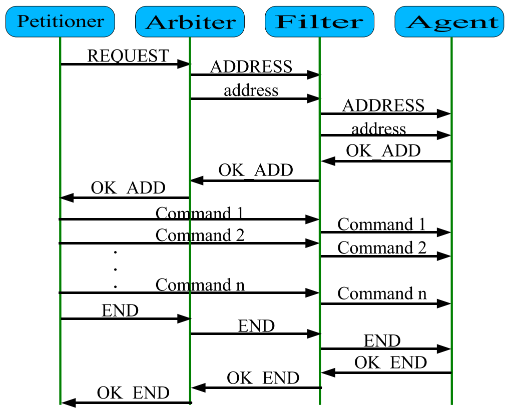


Figure 3.5: Permission protocol

There are four different agents: the petitioner, the arbiter, the filter, and the agent that manages the corresponding problematic element. The petitioner agent is the one which wants to use the "problematic" element. This agent must request permission from the arbiter agent before using the element ("REQUEST"). So, each agent that wants to use a "problematic" element has to request permission from the specific arbiter agent.

The arbiter agent maintains a queue of the requests that the agents have made and gives permission following a FIFO strategy. When the permission is given, the agent can use the element until it finishes its task.

When the arbiter agent deals with the request of a certain agent, it sends one message with the address of the petitioner agent to the filter agent indicating which agent has permission to use the element (“ADDRESS”, address). When the filter agent receives that message, it sends the same message to the agent that manages the element to inform it about which agent is sending it commands and to which one it has to send the responses of the commands executed.

When the agent that manages the element receives the address of the petitioner agent, it sends a message to the filter agent confirming that it is ready to receive commands (“OK_ADD”). The filter agent sends another message to the arbiter agent confirming the response of the element’s agent. And finally, the arbiter agent sends a message to the petitioner agent confirming that it has the permission to use the element.

From this moment, the petitioner agent can send as many messages as it wants until it finishes its task (“command n”). The messages must be addressed to the filter agent, and this agent sends them to the element’s agent if they are not dangerous for the robot. So, the filter agent works as a safety agent in charge of maintaining the integrity of the robots. If the command sent to the filter agent is conflictive, the filter agent doesn’t send it to the element’s agent, and sends a notification of error to the petitioner agent.

When the petitioner agent finishes its task, it sends a message to the arbiter agent indicating that it has finished using the element (“END”). The arbiter agent sends a new message to the filter agent to inform it that the last agent has finished and that it must wait for a new agent petition. The filter agent sends a message to the element’s agent to confirm that the agent has ended its task. The agent’s element sends a new message confirming that it has received the message from the filter agent (“END_OK”). This one sends a message to the arbiter as confirmation. And the arbiter sends a message to the petitioner agent to confirm that it does not yet have permission. Now, the arbiter agent takes the next petition from the queue or waits until a new one arrives.

With this approach, when one agent gains the access to the element, it has this access until it finishes its task. For the moment, for current applications this approach is enough. But in future, an important extension could be to introduce a maximum time for performing the tasks. Once this time is finished, the agent must join the queue again to regain permission. Or another possibility could be to add a system of priorities in the requests to the arbiter agent. Thus, if an agent has an urgent task to do, it can get the access temporarily returning to the queue the agents that in that moment are using the element. In that way, they are able to use the element immediately and do not have to wait to get the permission. Or even, a mixture of the two techniques would be more profitable.

3.3 Implementation of the architecture

For the implementation of the architecture described, two approaches were suggested: on the one hand, an object-oriented approach using the Java RMI, and on the other hand, an agent-oriented approach using any tool for developing multiagent systems.

Both approaches assume that the middleware must be implemented using Java. Since ARIA and Saphira are implemented in C++ and the middleware in Java, it is necessary to use a mechanism to integrate these different programming languages. For this reason, Java JNI (Java Native Interface) [206] has been used, so that C++ code can be managed, like the robots’ libraries, in Java programs.

JNI is a programming framework that allows Java code running in the Java virtual machine to call and be called by native applications (programs specific to a hardware and operating system platform) and libraries written in other languages, such as C, C++ and assembly. The JNI is usually used to write native methods to handle situations when an application cannot be written entirely in the Java programming language. It is also used to modify an existing application, written in another programming language, to be accessible to Java applications.

In figure 3.6, an example of communication, from the applications layer to the call to the function that moves the element, is shown.

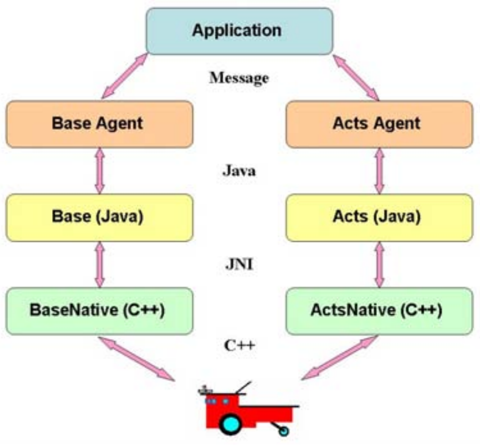


Figure 3.6: Different communication languages used in the Acromovi architecture

In this example, a generic application would try to move the base depending on the data that it perceives from the ACTS vision system. In order to do that, first it must to get the information from the ACTS by sending a message to the ActsAgent indicating which information the application needs. The ActsAgent calls the corresponding Java function from the Acts library which is in charge of communicating with the function of the C++ native library ActsNative that performs the action required. At this point it is when the Java JNI plays its role. The function in the ActsNative library simply calls to a function implemented in the robots' libraries. The result of the execution of the function with the desired value returns to the application layer in opposite direction. Once the application has this information, it sends another message to the BaseAgent which performs the same process once more in order to arrive at the robots' libraries and physically move the robot.

The two approaches cited before are described below. In the case of the agent-oriented approach two different implementations with different multiagent development tools are shown. Finally, a time performance is exposed to demonstrate that the approach followed is the best for the objectives of this thesis.

3.3.1 Implementation using software components: RMI

The first option was to use the Java RMI to create the middleware layer of the Acromovi architecture. The Remote Method Invocation (RMI) is the Java mechanism that allows a procedure (method, class, application, ...) to be invoked for remote object communication.

RMI applications often consist of two separate programs, a server and a client. A typical server program creates some remote objects, makes references to these objects accessible, and waits for clients to invoke methods on these objects. A typical client program obtains a remote reference from one or more remote objects on a server and then invokes methods on them. RMI provides the mechanism by which the server and the client communicate and pass information back and forth, the Stub/Skeleton approach.

The stubs and skeletons allow the remote functions to be simulated locally at the moment of their invocation. As it can be seen in figure 3.7, the stub works as a simulator for each invoked function that belongs to the server implementation. The skeleton works as a simulator to receive the parameters from the client implementation.

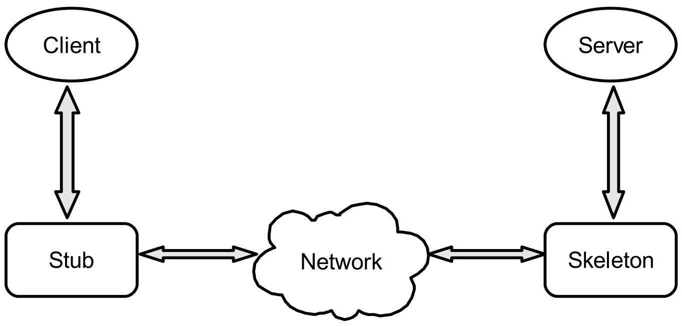


Figure 3.7: The Stub/Skeleton approach in Java RMI

Rather than making a copy of the client object in the server, RMI server passes a remote stub for a remote object to the client. This stub acts as a local representation for the remote object in the server, and basically is the remote reference. The client invokes a method on the local stub, which is responsible for carrying out the method invocation on the remote object of the server. The skeleton receives the invocation of the method from the stub and the parameters, and executes the corresponding method. Then, it returns the result of the method to the stub in the client.

This is used in the implementation of the architecture. The main idea is that each robot has a set of servers which give access to the resources of the robot, so that the applications, in the role of clients, can access these resources. So, in each robot there are a server for each element, thus, there is a server to manage the gripper, one to manage the movement of the robot, one for the camera, ... Moreover, there are other types of more advanced servers that are in charge of the navigation and localization of the robot. These servers are the ones which communicate with the lower layer of the architecture (ARIA + Saphira) by means of JNI.

Each one of these servers belongs to the robot where it is running, but they must be accessible by all the clients by means of RMI. Thus, the clients could be a PC controller or any other robot that executes a client. With this vision, it is easy to share the resources of a specific robot among the team, because all the robots have access to the features of all the other robots. Moreover, each robot is independent of the other, even so it is easy to create cooperative algorithms where more than one robot can take part.

An example of client program with this approach where actions are executed in two different robots is shown below:

```
Robot1.base.move(10); /* Move Robot1 10 meters */
Robot2.gripper.close(); /* Close Robot2 gripper */
if (Robot1.sonar.getClosestRange() <= 2) Robot2.base.move(2);
/* If Robot1 closer than 2 meters to Robot2, move Robot2 2 meters */
```

3.3.2 Implementation using a multiagent system

Initial implementation: MadKit

This second approach involves the implementation of the middleware layer using a tool for the development of multiagent systems. This option has been chosen due to the advantages that it offers the agent compared to the objects, making the agents more appropriate for the implementation of concurrent and distributed system, as has been shown in section 2.1.

The agents has two important capabilities, they are able to perform autonomous actions, that is, they can decide by themselves what they need to do in order to satisfy their own objectives, and they are able to interact with other agents to cooperate, collaborate, negotiate, ... [368]. So, they can be seen as autonomous and intelligent objects that are equipped with the capacities of knowledge and reasoning to satisfy several objectives.

In this case, the *MadKit* (MultiAgent Development Kit) [153] multiagent systems programming tool has been selected for the implementation of the architecture. MadKit is based on a organisational model called “Aalaadin”, based on three concepts: agent, group and role. MadKit also implements the community concept [114]. The relationship among these concepts can be seen in the figure 3.8.

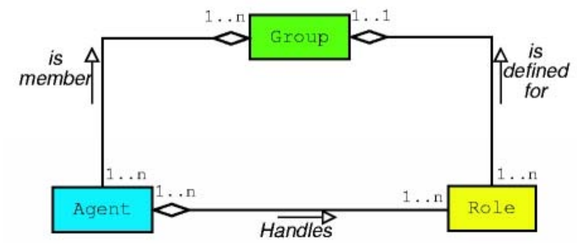


Figure 3.8: Aalaadin organizational model.

An agent is an active organization of communication that plays roles within the group. A group is a set of agents with similar functionality. Each agent can belong to one or more groups. A role is an abstract representation of the function or service of the agent. Each agent can have several roles, and each role is defined by a group. On the other hand, a community is a group of interconnected MadKit kernels. Each kernel acts as a node in a distributed environment. Thus, all the groups that are created must belong to one community [263].

Since the team includes seven mobile robots, the middleware layer is formed by a community of agents called “Seven Dwarfs”, that can be seen in figure 3.9. In this community, one active robot defines a group of its elements, which are represented and managed by agents. These agents communicate with the native layer of the global architecture by means of JNI.

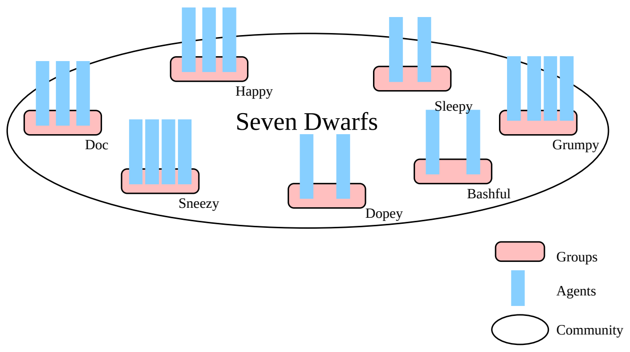


Figure 3.9: Structure of the "Seven Dwarfs" community

Initial implementation: Jade

The problem of MadKit is that it does not fulfill some practices that are "standard" in the domain of multiagent systems. The most important problem is the fact that MadKit does not use the FIPA specifications for their interoperability. So, a new multiagent system programming tool has been selected for the implementation of the middleware layer. This time, the multiagent tool chosen was JADE (Java Agent DEvelopment Framework) [193].

JADE is a software framework designed to develop agent-based applications in compliance with the FIPA specifications for interoperable intelligent multiagent systems. It is a software framework fully implemented in Java language and simplifies the implementation of multiagent systems through a middleware and through a set of graphical tools that supports the debugging and deployment phases [43]. The JADE middleware implements an agent platform for execution and a development framework. Also, it provides some agent facilities such as lifecycle management, naming service and yellow-page service, message transport and parsing service, and a library of FIPA interaction protocols ready to be used [43].

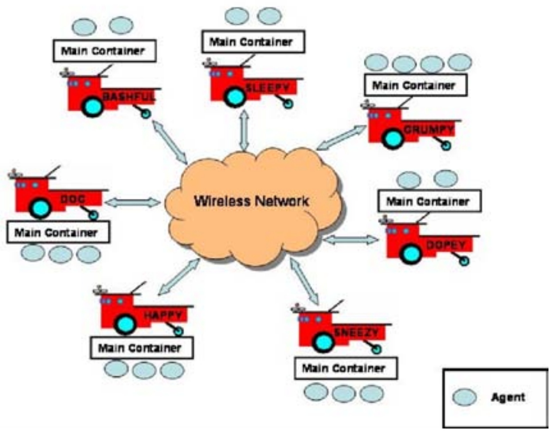


Figure 3.10: Structure of the Acromovi architecture implementation

The Acromovi architecture has been implemented following the JADE specifications. Thus, each robot involved in the architecture is a main container.Each one works as a host of a distributed network. In each container, there is a set of agents. These agents are created in execution time depending on the robot configuration, and each one represents one of the elements that at that moment the robot has active. All this, can be seen in the figure 3.10.

3.3.3 Architecture performance

Finally, in order to determine how good the architecture is, a time performance is shown. In this performance, the robot must realize the “test of the square”, that is, its movement must draw a square of 1m x 1m. In the test, the square is repeated 5 times.

Three different ways of communication have been tested. The first one consists of an implementation of the architecture using software components. These components communicate among themselves by means of the Remote Method Invocation (RMI) method. The second way is an implementation of the architecture using the MadKit multiagent system. And the third one is the implementation of the architecture described in this paper, using the JADE system.

The method for getting the times is the following; there is a client, a server and a controller for the base. The client communicates with the server to indicate to it which movement it should make. When the server receives the order from the client, it communicates with the controller to physically move the robot. The communications, in the case of RMI, are made by means of remote function invocation, and in the case of the MadKit and the actual architecture with JADE, by means of messages.

Also, different configurations relating to the place where the interactive entities are have been tested. Thus, in the cases of the MadKit and the actual implementations, the intra-platform and the inter-platform configurations have been tested. In the case of RMI, only the intra-platform configuration has been tested. In the intra-platform configuration, all the entities are in the same robot or computer. In the inter-platform configuration, there is a robot or computer with the client, and another one with the server and the controller. The information that is of interest for the performance is the transmission time between the client and the server.

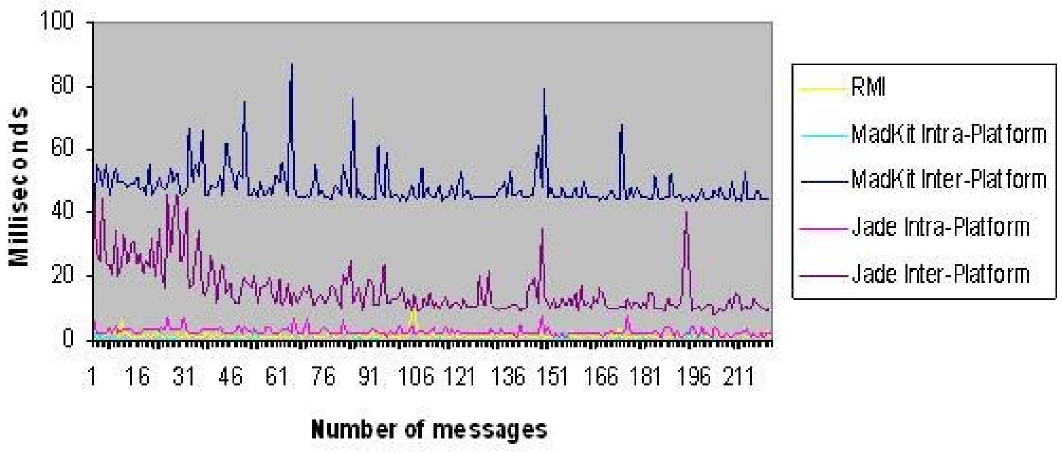


Figure 3.11: Timing performance among different implementations of the Acromovi architecture

A graph comparing these times can be seen in figure 3.11. It is important to note that the times shown in the graph are the time of sending the message from the client to the server plus the time of the answer from the server to the client. As can be seen, in the intra-platform configuration all the systems have a similar time. That is because JADE and MadKit internally use the RMI method when the messages are sent in the same platform. The differences come from the inter-platform configuration, in this case the actual configuration is better than the one implemented with MadKit.

3.4 Architecture extension for mobile manipulators

This section describes an extension added to the Acromovi architecture to manage an arm over a mobile robot [269]. Mobile manipulation involves the most important key issue in robotics: integration. While hardware integration seems to be nearly solved due to the increasing dominance of PC-compatible systems, software integration is still a challenge, since a lot of issues arise with the variety of operating systems, device drivers, application libraries, and programming languages which need to be merged in any real-world robotic system.

In recent years, there is an emergent interest in combining the two most important fields of robotics, the control of static manipulators and the navigation of mobile robots. By means of the mobility of the platform, there is an increase of the workspace of the manipulator. Also, having a manipulator on a mobile robot there is an increase in its operational capability and flexibility. Such systems allow robots to perform the most common missions of robotic systems which require both, locomotion and manipulation abilities. So, they seem particularly suited for field and service robotics [38].

Many different control systems for mobile manipulators have been developed. In [297], a control architecture suitable for mobile manipulation is explained. This architecture combines existing techniques for navigation and mobility with a flexible control system for the manipulator arm.

In [360], the authors implement a behaviour-based controller over a mobile manipulator to make it possible for the robot to open a door. The key issue for achieving such behaviour is cooperation between robots function sub-systems, such as the locomotion control system, the manipulator control system and the sensor systems.

Other approaches to the control of a mobile manipulator take into account neural networks. In [209], a neural network-based methodology is developed for the motion control of mobile manipulators subject to kinematics' constraints.

A task-oriented framework for the dynamic coordination of mobile manipulator systems has been implemented in [183]. The dynamic coordination strategy developed is based on two models concerned with the effector dynamic behavior, and the robot self-posture description and control. The effector dynamic behavior model is obtained by a projection of the robot dynamics into the space associated with the effector task, while the posture behavior is characterized by the complement of this projection.

Finally, other possible approach is to use multiagent system as in [110]. This work presents a multiagent system approach to a service mobile manipulator robot that interacts with human during an object delivery and hand-over task in two dimensions. The agents of the system are controlled using fuzzy control. The functions of the fuzzy controllers are tuned by using genetic algorithms.

In [207], a multiagent system architecture for a modular mobility enhancement system is presented. The system consists of one or more multiple mobile robotic platforms and a set of user defined application modules, such as chairs, tables multifunctional chairs etc. All these modules can be inexpensive everyday life items, which become functional when a mobile platform is properly attached to them. This way the system can act as a wheelchair, as a carrier of objects, or even as a walker.

Concerning vision, in [71] a new methodology to achieve vision-based control of a mobile manipulator is presented. This methodology makes it possible for the mobile manipulator to reach a required three-dimensional target position and orientation. The objective is to apply the non-holonomic degrees of freedom represented by two independently driven wheels together with the holonomic dexterity of the on-board arm in order to bring about a desired positioning objective for the arm.

Other work that presents a visual control is [325]. In this project, a high-precision visual control method for mobile manipulators is presented. That method makes efficient use of all the system's degrees of freedom, which reduces the required number of actuators for the manipulator.

Finally, the arm must be able to perform manipulation tasks. In [188], a framework for visually guided manipulation tasks is presented. The complexity of such tasks determines the precision and degrees of freedom controlled which also affects the robustness and flexibility of the system. The proposed approach results in a system which can locate and grasp a certain object.

3.4.1 Acromovi mobile manipulator extension

The general purpose is to add to the Acromovi architecture the capability of managing new components, an arm with all its accessories, and make feasible the cooperation between the arm and the mobile base, creating a mobile manipulator. The mobile manipulator used is formed by the combination of a Mitsubishi PA10 arm and an ActivMedia Powerbot mobile robot, as explained in section 1.2.1.

For the design of the framework, the characteristics of the two systems that constitute the mobile manipulator have been considered, the mobile base and the manipulator. The framework maintains the characteristics of the two sub-systems independently; while at the same time provides a basis for cooperation by means of the multiagent system. Also, the framework gives access to the accessories mounted on each sub-system making it possible to coordinate all the elements to perform a certain task.

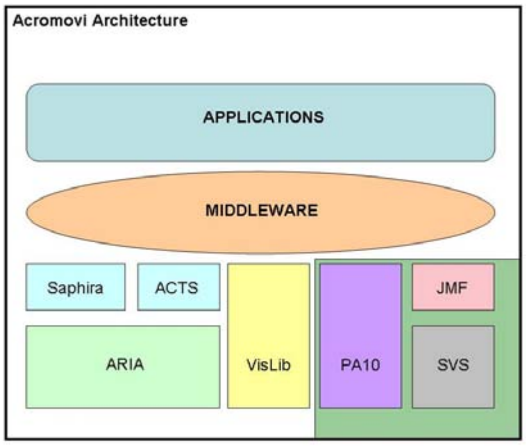


Figure 3.12: Extended Acromovi architecture

As in the case of the Vislib or ACTS, it is necessary to subsume new native libraries in the lower level of the global architecture. These native libraries are the libraries that permit the access to the arm and its accessories. This can be seen in figure 3.12.

The PA10 library is a library designed to develop application programs for the operation control of the robot, and moreover, is an interface library to ease the operation of all the functions for the motion control of the robot. It is written in C language. To manage the stereo, the C++ library Small Vision System (SVS) [185] is available. The SVS consists of a set of library functions for performing stereo correlation. Programs may call these functions to compute stereo results on any images that are available. It is written in C++.

These two libraries need to transfer their functions and information that return to Java. For this reason, the Java Native Interface (JNI) is used . In the case of the PA10 library, there is a wrapper that transforms the functions of this library to Java functions. For the vision system, there is a wrapper for the SVS, and it is able to deliver stereo 320x240 color images at 20 frames per second.

The stereo provides information that can be used to control the arm with a Java program. The acquisition of the images need the libraries provided with the cameras. Then these images are used to create the video stream using the Java Media Framework (JMF) [148]. One major advantage of using JMF is that is able to seamlessly transmit video over the network, using the RTP protocol [295]. With JPEG compression, the system is able to deliver 20 fps of stereo images through a wireless network (802.11b) to the university network, without noticeable loss of quality and response.

To control these new libraries, it is necessary to include new agents/components to the middleware layer. In more detail, in the lower level, as can be seen in figure 3.13. These new components are in charge of serving the petitions of the other agents of the system to the element that they manage.

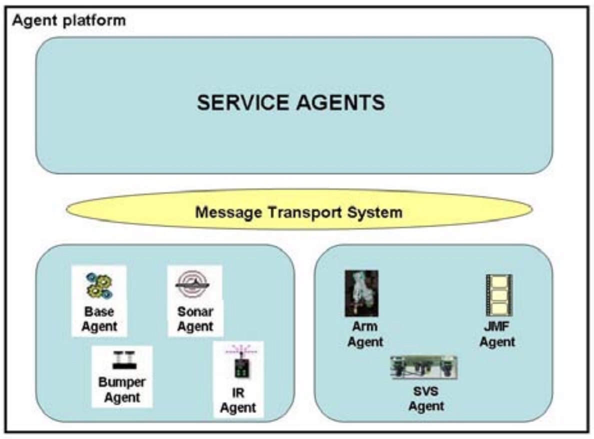


Figure 3.13: Mobile manipulator middleware

Thus, the first agent developed is the agent that moves the arm. The communication between the upper agents and the native library of the arm to move it is made possible by this agent.

In order to integrate the images of the stereo par with the agent architecture, a new agent has been implemented, the SVSAgent. This agent serves the images that are acquired by the stereo to the system, and offers all the facilities included in the SVS library.

Finally, the JMFAgent is capable of creating a video stream through the images that serve the SVSAgent. This video frame can be transmitted over the network without an elevated cost due to the jpeg compression. Also, this agent is able to execute some functions of image processing.

As can be deduced, each of the systems that form the mobile manipulator, the mobile base and the manipulator arm, are considered independently, but allowing their cooperation on the applications layer of the architecture to solve possible problems of global redundancy or control sharing.

It is the responsibility of the programmer to adapt the tools that the architecture offers to a mobile manipulation system. Also, it is possible, because of the scalability of the architecture, to implement a new application that makes possible the manipulation of the whole mobile manipulator, and then to implement that application as a new agent of the upper layer, allowing the agents of the architecture and other applications to access the facilities of the mobile manipulator.

3.5 Architecture extension for the Linuxbot robots

Finally, in this section another extension to the Acromovi architecture is described. This time, the extension consists of the incorporation of a new type of mobile robots to the general architecture, the Linuxbots, that were shown in section 1.2.1.

Linuxbots already have their own programming and control architecture. Thus, the followed process has been to extend the Acromovi architecture in order to integrate the Linuxbot architecture, so that the new mobile robots can use the Acromovi architecture. With this aim, it has been necessary to make a change in the approach of the architecture, and modify it using concepts from object-oriented programming, like superclasses, interfaces and inheritance.

Due to the simplicity of the Linuxbot robots, there are only two basic systems to take into account in the integration. On the one hand, it is the base of the robot that allows the robot to move through the environment, and on the other hand, it is the infrared frontal sensors, which allow the robot to detect if there is anything in front of it. Thus, following the design of the architecture, it was only necessary to create two new Java libraries and agents to manage these new systems.

From the control library of the Linuxbots, and with the help of JNI, the two new Java libraries, “Base_Linuxbot” and “IR_Linuxbot” were created. The first one to manage the base of the robot with the movements and the internal state, and the second one to manage the infrared sensors.

From these two libraries developed, it was easy to create the new agents, “BaseAgent_Linuxbot” and “IRAgent_Linuxbot”, which are in charge of the managing of the two components of the robot in the multiagent system of Acromovi architecture. To create these new agents, the JADE multiagent systems tool has been used. In the figure 3.14, the new libraries and agents can be appreciated, just as the mechanisms that allow their inter-connection with their upper and lower layers.

Once the classes and agents were created, and after discovering that many of the functions and procedures in the classes refering to the management of the base, it was decided to use concepts of object-oriented programming like superclasses, interfaces and inheritance in order to create a more functional architecture.

From the classes “Base_Pioneer”, to manage the Pioneer robots, and “Base_Linuxbot”, to manage the Linuxbots robots, it was possible to create an interface called “Base”. Base contains all those parts that are common between the two classes and which are necessary for a new mobile robot to be incorporated into the Acromovi architecture, like movement functions, or functions to consult the internal state of the robot.

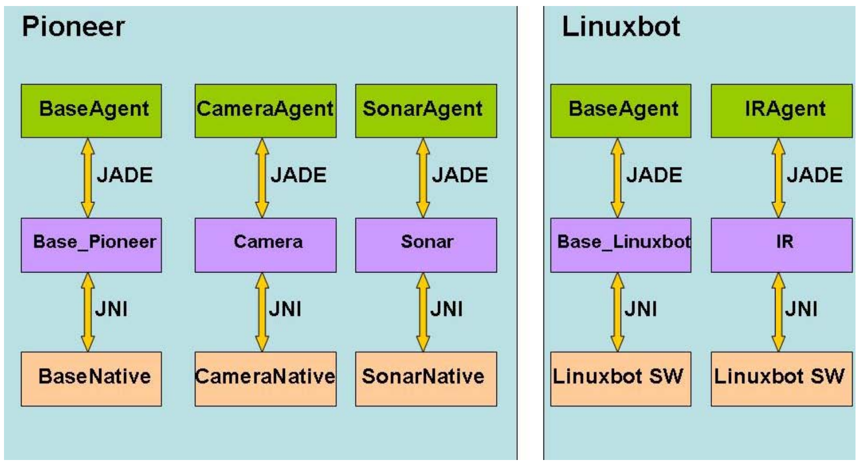


Figure 3.14: New agents and libraries for the Linuxbots robots

The new interface is used by BaseAgent, so that the access by part of the agent is transparent and always equal, regardless of the mobile robot that is working. In the case of the infrared sensors, a similar approach was followed.

In the figure 3.15, it is possible to see the relationship between the classes and the connexion with the agents in the new philosophy of the architecture.

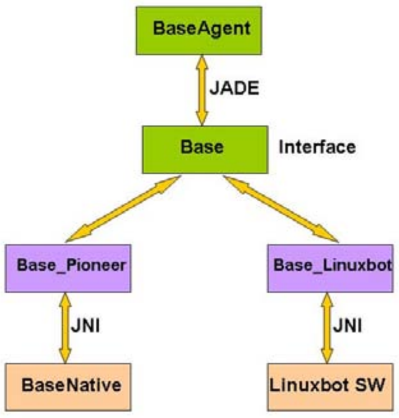


Figure 3.15: Example of interface communication for BaseAgent

Chapter 4

Cooperative navigation using the optical flow

Navigation by means of the optical flow technique and time-to-contact calculation is explained here. A new point of view is applied, extending the navigation from one robot to a group, the robots cooperating among themselves to navigate in a safe way using the techniques described below.

4.1 General overview

Think about a mobile robot navigating through a corridor in an office environment. The robot carries a camera to collect data about the environment and avoid possible obstacles in its way. This is the common scenario in mobile robots research.

While moving, the robot is taking images of the environment which allows it to calculate an estimate of the image motion. This estimate is of great importance in order to detect the objects in front of the robot and avoid them, as was seen in section 2.2.

In order to estimate the motion of an object in the scene, one of the techniques most used is the calculation of the optical flow, which is the distribution of the apparent velocities of movement of the brightness pattern in an image, and arises from the relative movement of some objects and a viewer [162]. As the robot is moving around, although the environment is static, there is a relative movement between the objects and the camera onboard the robot.

The visual information coming from the camera of the robot is processed through the optical flow technique, which provides the information the robot needs to safely navigate in an unknown working-environment.

From the optical flow generated in this way the robot can infer how far an object present in a scene is. This information is embedded in the time to collision (or time to crash) calculated from the optical flow field [322].

The purpose of the cooperative application implemented is to use the technique of optical flow so that a mobile robot equipped with a camera can navigate in a secure way through an indoor environment being able to avoid obstacles in real-time. This approach is then extended so that a group of robots can navigate in a secure way, but in this case only one robot has a camera and the information from the environment must be distributed among the team in such way that all the robots can navigate.

In order to implement such an application, the Acromovi architecture, that has been explained before, has been used. One of the features that makes Acromovi architecture suitable for the implementation of this application in that it allows the resources of each robot tobe shared among the whole team. In this case, the information received from the camera and computed as optical flow and time-to-contact is shared in such way that all the robots in the team can navigate without crashing into any obstacle.

So, the main objectives include the use of vision libraries for capturing and processing image frames, performing all the calculations as fast as possible, to implement simpler algorithms to perform the optical flow and time-to-contact calculations, and to include all these aspects inside the Acromovi architecture. The first objective implies the use of libraries such as Java Media Framework (JMF) for capturing images. The second objective is the most important to guarantee to the robot the capability of real-time response. The third objective is common in software development, and insures the accomplishment of the second objective. Finally, the fourth objective involves the use of the Acromovi architecture giving support for all the application.

For years, the problem of processing image sequences for calculating the optical flow has been studied for vision-based mobile robot navigation [72, 87, 101, 141, 145, 320, 319, 323, 322]. In some of these cases a specialized hardware is used [72, 87, 101], in others spatial cameras are used [87, 320], in others only small regions of the image are computed [72, 101, 141], or in others the image processing is done off board the robot in other computers [72, 101].

Also, in order to detect obstacles and avoid them there is a great number of studies. Nelson and Aloimonos [272] were the first to implement a simple obstacle avoidance algorithm using flow divergence for a camera mounted on a robot arm. Ancona and Poggio [10] use a 1-D correlation-based method for detecting motion and visual motion is only calculated where time-to-contact measures are to be taken.

Also, Coombs et al. [87] use normal flow in the central region of the camera view to recover flow divergence for real-time time-to-contact estimation, which is then used to decide whether to turn or to stop when collision is imminent. Finally, Santos-Victor and Sandini [318] use optical flow to achieve obstacle avoidance without the direct use of time-to-contact or flow field divergence. Obstacles are detected by observing obstructions in the expected flow field induced by the ground plane. More recent works include those of Sarcinelli-Filho [323, 322], or Souhila [333].

4.2 Background Theory

This section presents the theory about the techniques used in the implementation on the navigation tasks. This techniques are the optical flow and the time-to-contact.

4.2.1 Computation of the optical flow values

Optical flow computation consists in extracting a dense velocity field from an image sequence assuming that intensity or color is conserved during the displacement. This result may be used by other applications such as time interpolation of image sequences using motion information.

The optical-flow field used in this work is obtained by using formulas and constraints used by Horn and Schunck [162]. The equation that relates the change in image brightness at a point to the motion of the brightness pattern is:

$$I_x u + I_y v + I_t = 0 \tag{4.1}$$

where, $I(x, y, t)$ represents the value of brightness at time t of a point with coordinates (x, y) in the image plane. I_x, I_y, I_t represent the partial derivatives of I with relation to x, y, t respectively. u, v denote the components of optical flow along x and y respectively.

However, this equation, that is called "optical flow constraint equation", is not enough to determine u and v at any pixel from the image, so a new constraint should be included. Horn and Schunk use an additional constraint in which the sum of the Laplacians of u and v should be minimized. This constraint is called "smoothness constraint" [162]. This means that

$$\nabla^2 u + \nabla^2 v = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \tag{4.2}$$

must be minimized.

They approximate the values of the Laplacians as

$$\begin{array}{c} \nabla^2 u \approx 3(\bar{u} - u) \\ \nabla^2 v \approx 3(\bar{v} - v) \end{array} \tag{4.3}$$

where $\bar{u}$ and $\bar{v}$ are the average values of the optical flow components taken over 3x3 pixels square centered on the pixel under consideration. That can be calculated as follows

$$\begin{array}{c} \overline{u}_{i,j,k} = 1/6 \left\{ u_{i-1,j,k} + u_{i,j+1,k} + u_{i+1,j,k} + u_{i,j-1,k} \right\} \\ +1/12 \left\{ u_{i-1,j-1,k} + u_{i-1,j+1,k} + u_{i+1,j+1,k} + u_{i+1,j-1,k} \right\} \\ \overline{v}_{i,j,k} = 1/6 \left\{ v_{i-1,j,k} + v_{i,j+1,k} + v_{i+1,j,k} + v_{i,j-1,k} \right\} \\ +1/12 \left\{ v_{i-1,j-1,k} + v_{i-1,j+1,k} + v_{i+1,j+1,k} + v_{i+1,j-1,k} \right\} \end{array} \tag{4.4}$$

That is, the following mask is used in the calculus of the Laplacians

$1/12$	$1/6$	$1/12$
$1/6$	-1	$1/6$
$1/12$	$1/6$	$1/12$

The iterative solution for the values of u and v is given by:

$$\begin{array}{c} u^{n+1} = \overline{u}^n - I_x \left[I_x \overline{u}^n + I_y \overline{v}^n + I_t \right] / \left(\alpha^2 + I_x^2 + I_y^2 \right) \\ v^{n+1} = \overline{v}^n - I_y \left[I_x \overline{u}^n + I_y \overline{v}^n + I_t \right] / \left(\alpha^2 + I_x^2 + I_y^2 \right) \end{array} \tag{4.5}$$

where $n + 1$ is iteration under calculation, n is the previous iteration, and α is a weighting factor.

Finally, the values of I_x, I_y and It can be calculated as follows:

$$\begin{array}{l} I_x \approx 1/4[I_{i,j+1,k} - I_{i,j,k} + I_{i+1,j+1,k} - I_{i+1,j,k} + I_{i,j+1,k+1} - I_{i,j,k+1} + I_{i+1,j+1,k+1} - I_{i+1,j,k+1}] \\ I_y \approx 1/4[I_{i+1,j,k} - I_{i,j,k} + I_{i+1,j+1,k} - I_{i+1,j,k} + I_{i+1,j,k+1} - I_{i,j,k+1} + I_{i+1,j+1,k+1} - I_{i,j+1,k+1}] \\ I_t \approx 1/4[I_{i,j,k+1} - I_{i,j,k} + I_{i+1,j,k+1} - I_{i+1,j,k} + I_{i,j+1,k+1} - I_{i,j+1,k} + I_{i+1,j+1,k+1} - I_{i+1,j+1,k}] \end{array} \tag{4.6}$$

As can be seen, this algorithm only uses two different images $(t, t+1)$, thus very little quantity of memory is needed. Also, in the initialization it is necessary to provide the algorithm with the number of iterations to be performed (n). Depending on this number, the computational consumption will be higher.

Following the algorithm, the optical flow can be calculated. In figure 4.1, an example of this calculation is shown.

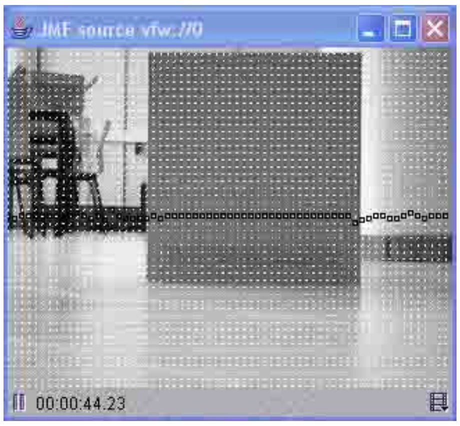


Figure 4.1: Optical flow field.

4.2.2 Computation of the time to contact values

A primary use of optical flow in robotics vision is collision detection, in particular time-to-contact computation. Using only optical measurements, and without knowing one's own velocity or distance from a surface, it is possible to determine when contact with a visible surface will be made [69].

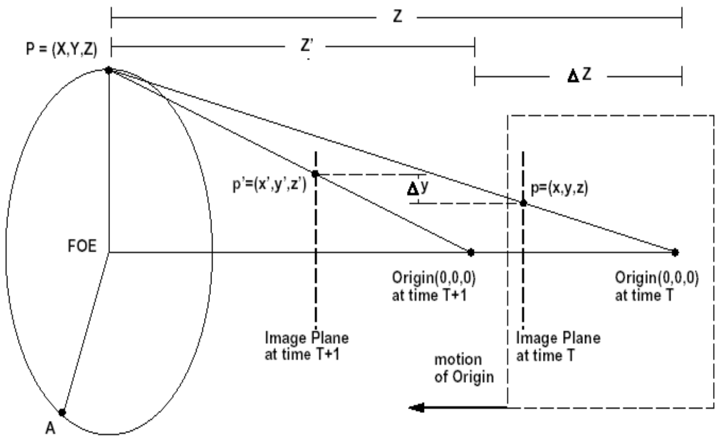


Figure 4.2: Optical geometry for time-to-contact.

A point of interest P at coordinates (X, Y, Z) is projected through the focus of projection centered on the origin of the coordinate system $(0, 0, 0)$. P is fixed in physical space and does not move. The origin/focus of projection however moves forward with velocity dZ/dt. If the camera is facing the same direction as the direction of motion, then this direction is what is commonly known as the "focus of expansion" (FOE), since it is the point from which the optical flow diverges. The image plane is fixed at a distance z in front of origin. For convenience, it is set to $z = 1$ (The actual value of z depends on the focal length of camera). P projects onto point p in this plane. As the image plane moves closer to P, the position of p in the image plane changes as well. Using equilateral triangle,

$$y/z = y/1 = Y/Z \tag{4.7}$$

and differentiating with respect to time and taking the reciprocals in both sides,

$$y/\dot{y} = -\left(Z/\dot{Z}\right) = \tau \tag{4.8}$$

The quantity τ is known as the time-to-contact. Note that the left-hand side contains purely optical quantities, and that knowledge of τ does not provide any information about distance and velocity, only their ratio. In figure 4.3, a example of the calculation that provides this method is shown .

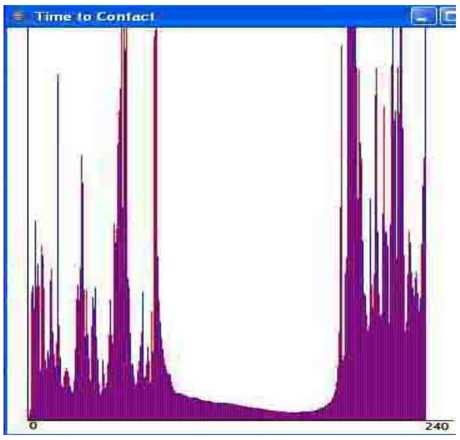


Figure 4.3: Example of time-to-contact graph.

4.3 Navigation of a single robot

This section comments on the method and implementation performed in order that a single robot with a camera mounted on it can navigate through an unknown environment.

4.3.1 Control system and navigation

Based on the optical-flow field and the time-to-contact values obtained as described above, the sensing subsystem delivers that information to the control subsystem. As can be seen in figure 4.3, the time-to-contact is divided in distinct stripes in the visual field of the robot. The values of these stripes are considered to control the heading angle of the robot.

The stripes of the time-to-contact correspond to the columns of pixels in the image. In each of these columns, the value corresponds to the minimum value of the time-to-contact for that column, thus obtaining the most dangerous value in this place. So, 240 values of the time-to-contact are available.

In order to calculate the evasion manoeuvre, two options have been defined. The first one consists of dividing the possible situations that can occur in the vector of time-to-contact in three general situations. These situations are:

- *Imminent Collision*, when an object is very near to the robot, that is, the values of the time-to-contact in the middle of the image are less than the threshold. In this case, the action the robot takes is to go back 10cm, to rotate 180 degrees in one direction and to move forwards again.

- *Side Obstacle*, similar to the previous one, but in this case the object is in the left or right side of the image. In this case, the action the robot takes is to rotate 15 degrees in the opposite direction and to resume moving forwards.

- *Normal Situation* is characterized by the situation where no objects with low time-to-contact are detected in the image frame, that is, all the values are above the threshold. This means that none of the objects in the visual field of the robot are close enough to deserve an abrupt manoeuvre. When this is the case, the robot moves straight ahead.

In order to differentiate among the Imminent Collision and Side Obstacle situations, three different areas must be defined to separate the stripes of the time-to-contact that defines each of these situations. Thus, the left Side Obstacle situation includes the area between the stripe 0 and 80, for the Imminent Collision situation, the are between 80 and 160 stripes, and for the right Side Obstacle situation, the stripes between the 160 and the 240 ones. This can be seen in figure 4.4.

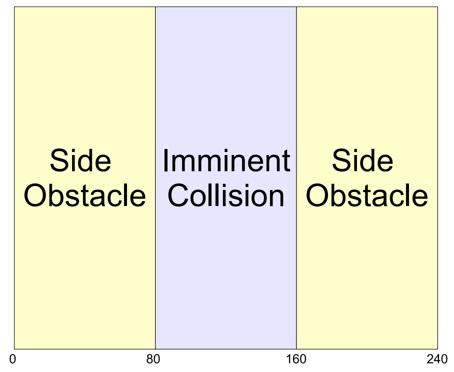


Figure 4.4: Distribution of the different situations of the time-to-contact

The second option tries to make the movements of the robot smoother. The path followed by the robot through the environment will therefore be also correspondingly smooth.

First, the 240 values of the time-to-contact graphic are grouped in groups of 10, segmenting the image in different regions that execute similar movements. The final result of such segmentation of movement is a vector of 24 values of time-to-contact, which are the mean of the 10 values that are grouped. Each value of the time-to-contact corresponds to the object closest to the robot in each angular interval of 2 degrees of the visual field of the robot (which is 48 degrees). Each angular interval of 2 degrees corresponds to a region whose width is 10 pixels, so that obtaining a vector of 24 times to contact is equivalent to using 24 CCD cameras, each one with a visual field of 2 degrees.

Such vector of time-to-contact is the sensorial information used to control the heading angle of the robot in this case. Knowing that the center of the image corresponds to the 0 degrees of rotation, the left side corresponds to positive angles until 24 degrees, and the left side to negative angles up to -24 degrees.

In this case the basic idea is to recognize which is the direction with more free space and to steer the robot in that direction. This direction is represented in the graph of the time-to-contact values as the region with the highest value.

The operation of the control system in this case is as follows. If all the values of the time-to-contact are over a defined threshold, the robot must move straight ahead. But if any of the values is under the threshold, in this case the robot must detect it and perform the evasive manoeuvre, if it is the case, to avoid the obstacle. Knowing the position of the obstacle in the image and the position which has more free space, it is easy to calculate the angle that the robot must turn in order to avoid that obstacle.

4.3.2 System Development

To give support for the system that has been explained, three new agents have been designed for the development of this application in the Acromovi architecture, the JMFAgent, the OpticalFlowAgent, and the ControlAgent. Each of these agents implements each of the modules described in previous sections. A diagram of the relation among these agents can be seen in figure 4.5.

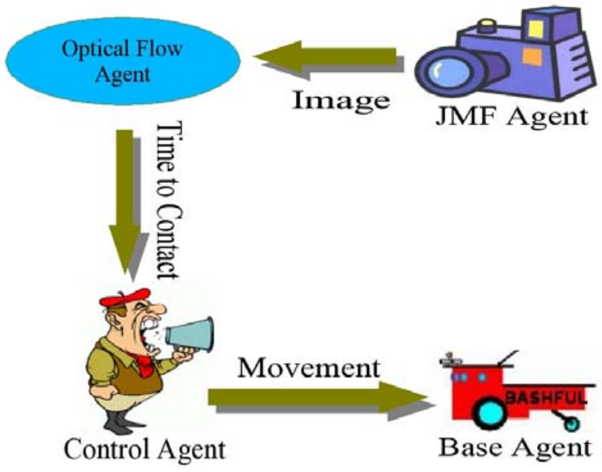


Figure 4.5: Interaction among the agents

The first agent involved in the process is the JMFAgent, which is in charge of capturing the input image frames. The video from the camera on board the robot is captured using the Java Media Framework (JMF). The agent produces an image frame of size 320x240 pixels and provides grey scale data of each pixel at the rate of 5 frames per second. In figure 4.6 there is an example of frame captured by the JMFAgent.

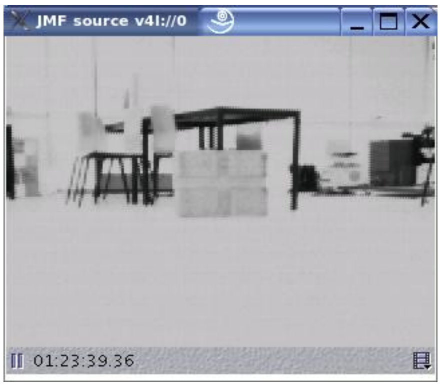


Figure 4.6: Captured image by the JMFAgent

Next, this image is sent to the next agent in the process, the OpticalFlowAgent. This agent performs two different tasks. Firstly, it calculates the optical flow derived from the new image, and then, it calculates, from the optical flow, the time-to-contact vector. In figure 4.1, there is an example of the optical-flow field and in the figure 4.3, the time-to-contact graphic.

Next, the OpticalFlowAgent sends the information corresponding to the time to contact to the ControlAgent, which implements one of the two strategies explained before. Also, this agent is the one which sends the different actions to the BaseAgent so that the robot can avoid an obstacle or can follow its path.

An important disadvantage of the system is that after any turning, the calculation of the time-to-contact values must be elapsed during one image frame. This turning can produce an abrupt change in the image frame, and therefore it can cause an incorrect optical flow field to be acquired or a very high magnitude which can result in low time-to-contact values and hence a wrong decision which can put the physical integrity of the robot in danger.

4.3.3 Results

In order to check the performance of the sensing and control subsystems discussed before, the robot is programmed to wander around the lab, avoiding all the obstacles it detects with both the methods explained.

The robot acquires image frames of 320x240 pixels continuously at an interval of 200 ms, the calculation of the optical flow vectors plus the calculation of the new heading angle is compatible with the rate of acquisition of images, thus showing that the use of optical flow for this kind of sensing is suitable.

An analysis of all the actions the robot has taken shows that it was effectively able to avoid the obstacles that appeared in its way, as expected, using the time-to-contact based sensorial information. In both cases the robot performs the tasks in a correct way. But, it is important to note that the second option is more precise in the avoiding of the obstacles and does not perform abrupt movements. Some examples of executions can be seen in the following figures.

The first one, in figure 4.7, corresponds to an experiment using the first strategy which divides the frame into three possible situations. In this case, a manoeuvre to avoid the obstacle in the situation “Side Obstacle” is shown.

The second experiment, in figure 4.8, corresponds to the “Imminent Collision” situation of the first strategy. It can be seen that when the robot arrives too close to the object, it turns 180 degrees and comes back.

Finally, an example of execution following the second strategy is shown in figure 4.9. In this case, it is possible to see how the robot when it approaches the obstacle, selects the direction that offers more free space to the navigation of the robot.

4.4 Cooperative navigation using the optical flow

In this section a new approach to the visual navigation described in the previous section is explained. In this case, it is not only a robot which navigates through the environment avoiding the obstacles. In this case, there are two robots navigating at the same time, but the approach that is described can be used to make more than two robots navigate at the same time.

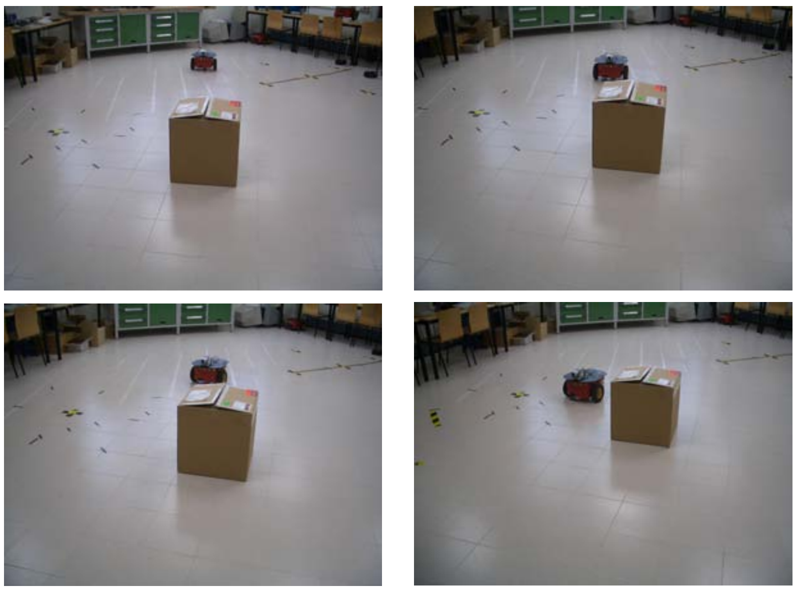


Figure 4.7: Strategy followed in the case of the “Side Obstacle” situation

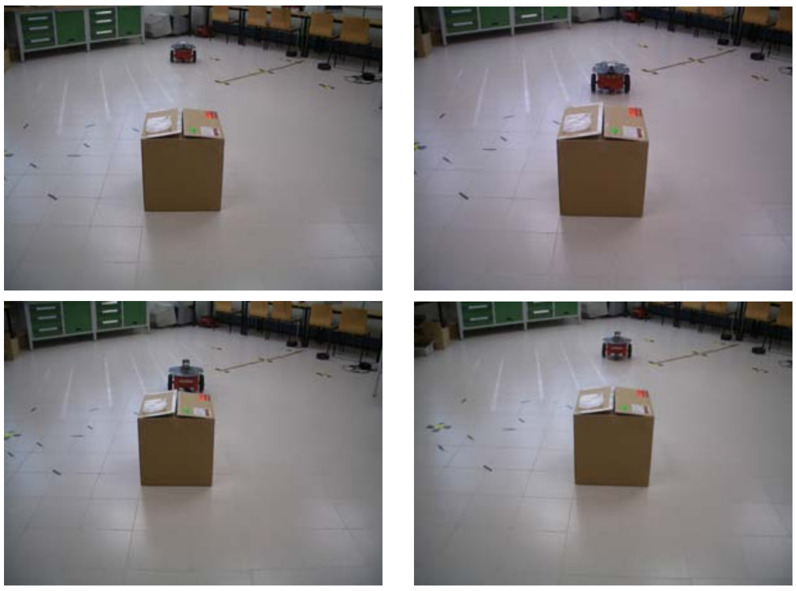


Figure 4.8: Strategy followed in the case of the “Imminent Collision” situation

4.4.1 Control system and navigation

This application allows a number of robots to navigate through an unknown environment in a secure way. The environment is restricted to corridors or environments like a corridor where the robots can move forward for long periods without encountering obstacles.

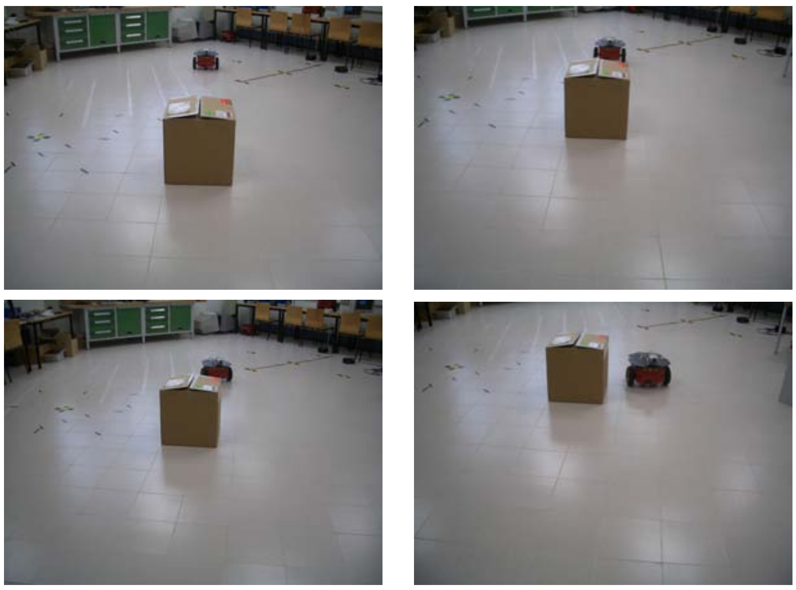


Figure 4.9: Strategy followed with the second strategy

Any number of robots can form the team but in this example only two robots are taken into consideration. One of these robots has a camera mounted on it, which allows the robot to compute the optical flow field and the time-to-contact values in order to avoid the obstacles that can appear in the way. The other robot does not carry any sensor that could give it any information about the environment.

The idea is that the information of the robot with camera, from this point the leader, can be shared in any way that helps the robot without a camera to navigate through the environment without crashing into any obstacle. For this reason, the robots must maintain a specific position each one with regard to the other, so that the leader robot can know the position of the "blind" robot and help it when an obstacle appears in the way. The positions of the robots considered for this application are shown in the figure 4.10.

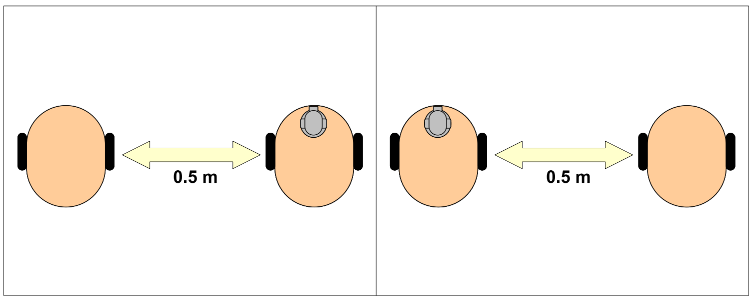


Figure 4.10: Positions of the robots at the beginning of the application

The two robots always move forwards in a straight line while the leader does not detect any obstacle for it or the the other by means of the optical flow and time-to-contact values, as was explained in section 4.2. In the moment that the leader detects any obstacle, an evasion manoeuvre must be made in order to avoid the obstacle and maintain the integrity of the robots.

Different situations can arise when detecting an obstacle. Depending where the obstacle is located, the robots will take different strategies for avoiding it. These strategies can be divided into two groups.

The most common situation is when there is enough space to avoid the obstacle and so the robots merely pass by one side or the other. Depending on which side there is more free space, the robots move to that side. This can be seen in figure 4.11.

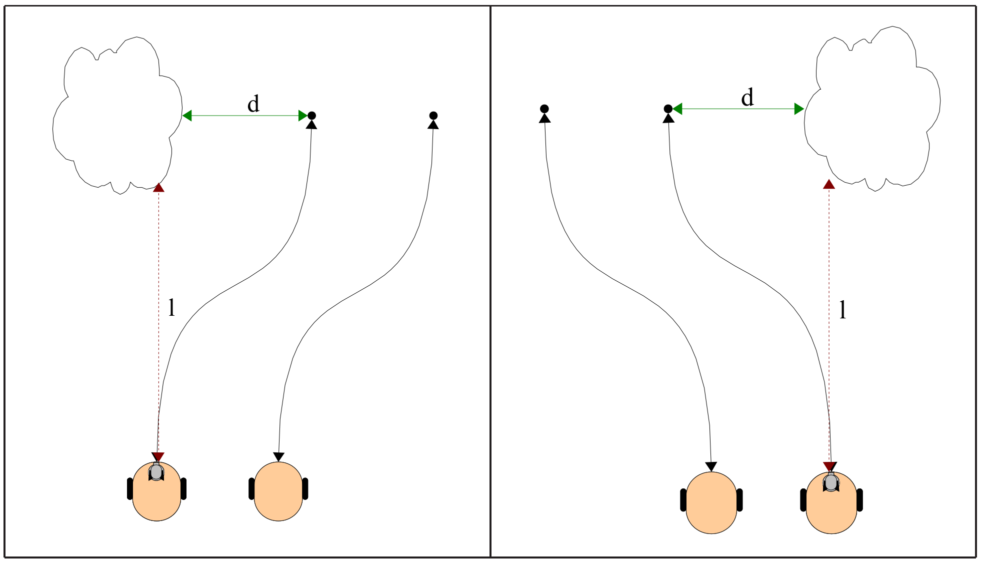


Figure 4.11: Maneouvres taken to avoid an obstacle passing the two by the left or the right side

The other situation occurs when there is not enough free space for the two robots to pass by one of the sides of the obstacle, then the robots must divide one on each side. This is represented in figure 4.12.

As can be seen in the figures, when an obstacle is in the path of the robots, the robots must modify their current path in order to avoid it. For this reason, the control system of the leader robot must calculate the point where the robots must move to avoid the obstacle. With the information of the time-to-contact and the data that the robot has at that moment, it is possible to calculate these points for both robots.

First of all, it is necessary to know the width of the obstacle and the free space that remains on both sides of the obstacle in order to choose one of the possible strategies seen before. The time that remains to contact with the obstacle in frames is known and indicated by the time-to-contact values, and the frequency at which each new frame is acquired from the system is also known. With this information it is easy to calculate the time to contact that remains by the following equation,

$$t = \frac{1}{f} \cdot ttc \tag{4.9}$$

where t is the time in seconds to crash with the obstacle, f the frame frequency, and ttc the number of frames from the time-to-contact.

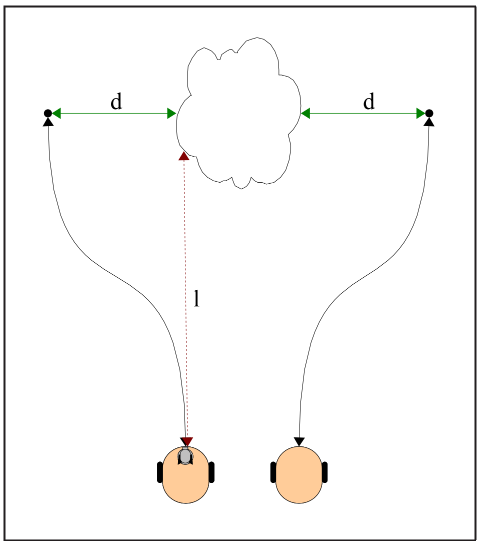


Figure 4.12: Manoeuvres to avoid an obstacle passing each robot by one side

Once the time to contact is known, and knowing the velocity of the robot, it is also easy to calculate the distance to the obstacle,

$$dist = t \cdot v \tag{4.10}$$

where t is the time calculated previously and v is the actual velocity of the robot.

Now, from the time-to-contact vector it is possible to calculate the width of the obstacle in the image. Simply, recognizing which values of the time-to-contact corresponds to the obstacle, counting the stripes that it fills and multiplying this value by 10 pixels grouped in each stripe, the width of the obstacle in the image is calculated, as shows figure 4.13.

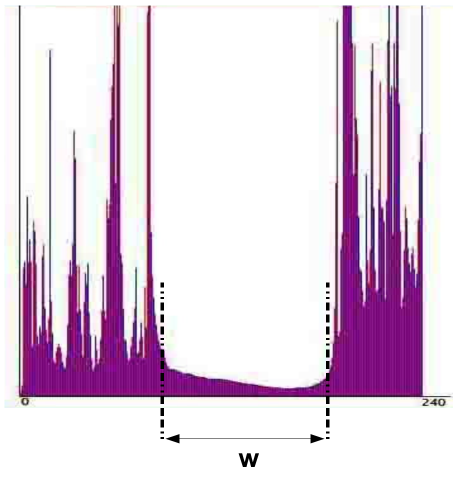


Figure 4.13: Width of the obstacle calculated in the time-to-contact vector

To determine the width of the obstacle in the real world, it is necessary to use some intrinsic parameters of the camera. Supposing that the robot has a “pin-hole camera”, the different elements in the scene are related as shows figure 4.14.

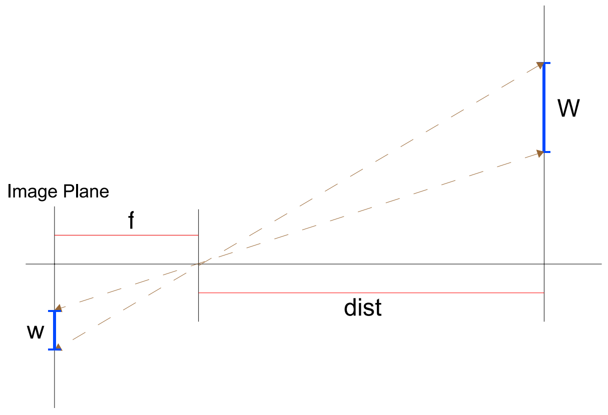


Figure 4.14: Pin-hole camera model

From the figure it is possible to extract information that allows the real width of the obstacle to be calculated. It is necessary to know three parameters, the focal length of the camera, the width of the object in the image and the distance from the camera to the object. With this information, the width of the object can be calculated as follows,

$$W = dist \cdot \frac{w}{f} \tag{4.11}$$

The free space remaining on both sides of the obstacle can be calculated following the same strategy. With that information, the robot has to decide one of the possible strategies and depending on that, it has to calculate the points to which the robot must move in order to avoid the obstacle.

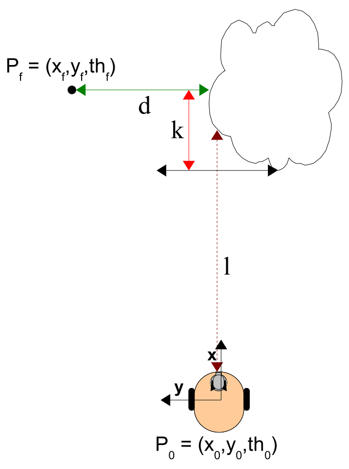


Figure 4.15: Calculus of the destiny point to avoid the obstacle

Depending on the strategy chosen to avoid the obstacle, different points must be calculated, but all of them follow the same calculation procedure. That is, knowing the actual position of the robot P_0, the distance to the object l, a parameter d, which indicates the separation between the robot and the object, and other parameter k, that indicates the vertical separation from the front side of the obstacle, the point to move the robot P_f can be calculated. All these elements are represented in the figure 4.15.

Also it is necessary to know which portion of the obstacle still remains on the side that the robot has chosen to move. Supposing that the robot moves in the center of the time-to-contact vector, simply counting the pixels that remain until the end of the obstacle and applying the equation 4.11, the size of the obstacle that remains in the way of the robot, O, is calculated. With this data, the final point to move the robot can be calculated as follows:

$$\begin{aligned} x_f &= x_0 + l + k \\ y_f &= y_0 + O + d \\ th_f &= th_0 \end{aligned} \tag{4.12}$$

The robot leader calculates these points for every robot that is present in the execution of the application. Depending on the strategy used, different points will be calculated for the robots, but all the points follow the same calculation.

Once the points for all the robots are calculated, the leader sends them to the corresponding robot in order that this robot can avoid the obstacle. When the robots receive this point, they must move to it. The strategy followed to move the robot from one point to another is to use *Bezier curves* [79], which allow them to move in a smooth way without great changes in the orientation of the robots. As can be seen in the figure 4.16.

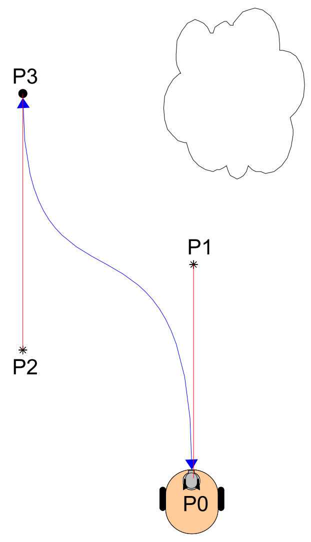


Figure 4.16: Bezier Curve to move from one point to another

Such curves can be defined with only four points: the endpoints $P0$ and $P3$ are given by the actual position of the robot and the end position; the control points $P1$ and $P2$ are chosen along the lines defined by the orientations of the robots.

The cubic Bezier curve is given by

$$P(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = at^3 + bt^2 + ct + P_0, t \in [0,1] \tag{4.13}$$

where the vectors a, b, c are defined as follows

$$\begin{aligned} c &= 3(P_1 - P_0) \\ b &= 3(P_2 - P_1) - c \\ a &= P_3 - P_0 - c - b \end{aligned} \tag{4.14}$$

and the points are defined in the follower robot's reference frame

$$P_0 = \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}, P_1 = \begin{pmatrix} x_0 + d \\ y_0 \end{pmatrix}, P_2 = \begin{pmatrix} x_f - d \cdot cos\theta \\ y_f - d \cdot sin\theta \end{pmatrix}, P_3 = \begin{pmatrix} x_f \\ y_f \end{pmatrix} \tag{4.15}$$

where the value of θ is the orientation of the final point with respect to the initial one, usually this value will be 0 in order to adopt the same orientation so that it can follow the same way, and the value of d can be chosen arbitrarily, but depending on this value, the curve to be followed by the robot is different. In this application it is calculated as,

$$d = \frac{2}{3}\sqrt{2}(\sqrt{2} - 1)\, \|x, y\| \tag{4.16}$$

The robot must follow this trajectory that the Bezier curve indicates in order to arrive at the end point. Assuming that the robot moves at a constant linear velocity v, to compute the angular velocity ω, the computation of the curvature κ is necessary since

$$\omega = \frac{v}{R} = v\kappa \tag{4.17}$$

where R and κ are the radius and curvature of the trajectory, respectively. The curvature of any parametric curve is calculated as

$$\kappa = \frac{x^{'}y^{''} - y^{'}x^{''}}{(x^{'2} + y^{'2})^{3/2}} \tag{4.18}$$

So, in order to move the robot from one point to the other and avoid the obstacle, the following velocities must be set

- Linear velocity: v
- Angular velocity: $\omega = v\kappa$

The robots move following the described curve to arrive at the end point. Once they have arrived at this point, they continue moving in a straight line while the leader robot is "seeing" the obstacle.

When the leader no longer perceives the obstacle, a manoeuvre to return to the original positions is made. This manoeuvre is similar to the one to avoid the obstacle. It is necessary to calculate a new Bezier curve with new points, calculate the angular velocity and move the robot until it arrives at the desired position. Once the robots have arrived at the position, they continue moving in a straight line until a new obstacle appears in the way.

4.4.2 System Development

In this case, the development for the two robots is taken into consideration. In the leader, the same agents that in the case of only one robot are present. However, it is necessary to change the ControlAgent so that it can take decisions about which strategy to use, both on the same side or make a bifurcation, and can calculate the points to which the robots must move in order to avoid the obstacle.

In the other robot, it is only necessary to add a ControlAgent that receives the point to which it must move from the leader and moves the robot to this point by means of a Bezier curve.

For this reason, a communication among the two robots is necessary. This communication is limited to the minimum and the two robots only communicate among them 3 times. The first one is when the leader communicates that the movement can begin. The second one is performed when the leader detects an obstacle and sends the point to the robot to which it must move. Finally, the third communication act is made when the leader no longer sees the obstacle and sends the point to the robot to return to the original position.

4.4.3 Results

Also in this case, in order to check the performance of the sensing and control subsystems discussed before, the robots are programmed to move in a corridor-following application, avoiding all the obstacles that they detect with the method explained.

The robots at the beginning are placed one besides the other with a distance of 0.5 meters. The robot with no sensing system can be at the right side or the left side of the leader, but the leader must know this position in order to perform the calculations.

The analysis of the actions that the robots take shows that they are able to avoid the obstacles that appear in their way. In all the possible variations the robots perform the tasks correctly. Two examples of executions can be seen in the following figures.

The first one corresponds to an experiment where the two robots can pass both on the same side of the object. The manoeuvre to avoid the obstacle can be seen in the sequence of images shown in figure 4.17.

Finally, this example of execution corresponds to the case in which it is not possible that both robots pass on the same side. In this case, the robots must perform a bifurcation and once the obstacle has been passed they must come back to the original positions. In the following series of images, this behaviour can be seen in figure 4.18.

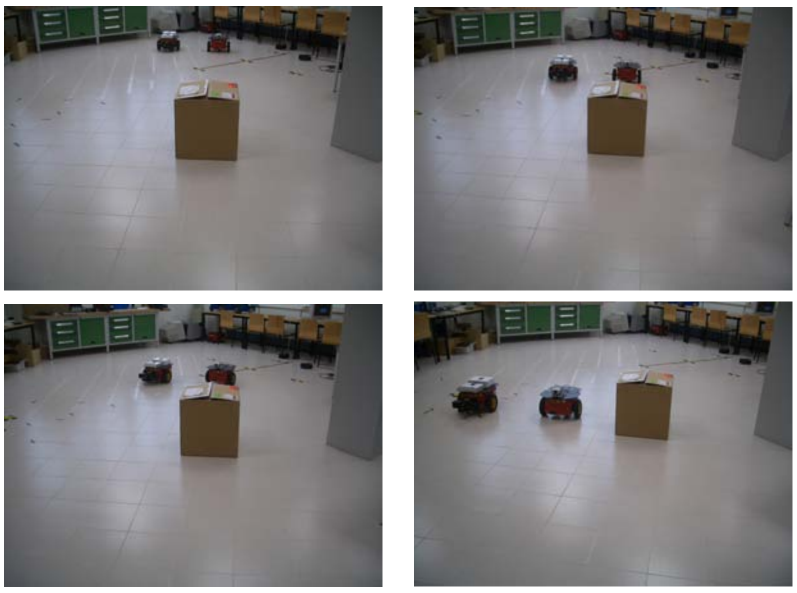


Figure 4.17: Strategy followed in the case that the two robot can pass by the same side

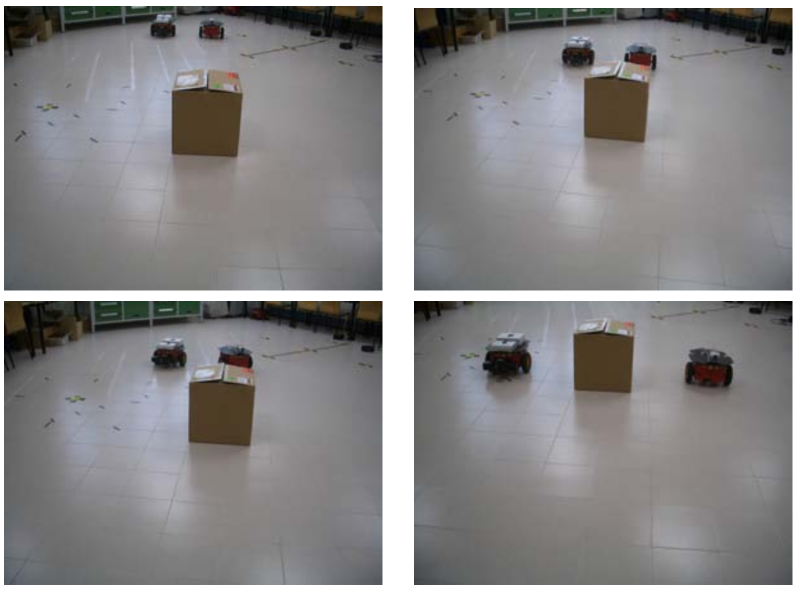


Figure 4.18: Strategy followed in the case that the two robot cannot pass by the same side

Chapter 5

Cooperative formations of multiple mobile robots

This chapter explains how to create and mantain a multirobot formation. The formation is made up of a group of robots in which only one has a laser system which is used to navigate through the environment and only one has a camera to localize the other robots and is used to mantain the formation. The cooperation among the robots is also explained in this chapter.

5.1 General overview

As seen in section 2.3, coordination among a group of robots can be very useful for many applications. One of the most important tasks is in motion coordination, is how to move a team of robots in an ordered way, such as maintaining a predefined formation.

In this way, one of the first systems that used specific formations for a particular task were scout teams of the US Army. In moving quickly down roadways for instance, it is often best to follow one after the other. When sweeping across desert terrain, line-abreast may be better. Furthermore, when scouts maintain their positions, they are able to distribute their sensor assets to reduce overlap. Army manuals list four important formations for scout vehicles: diamond, wedge, line, and column.

Balch and Arkin [28, 30] used homogeneous, reactive, non-communicating agents to study formation maintenance in autonomous robots. The robots' goal is to move together in a military formation such as a diamond, column, or wedge. They periodically come across obstacles which prevent one or more of the robots from moving in a straight line. After passing the obstacle, all robots must adjust in order to regain their formation.

Formations are very important in the field of multiagent robotics, in order for the robots to roam together to complete a certain task they need to form a chain and maintain this formation [160, 174, 276]. Control of robot formations has received much attention in the past years.

One of the first approximations to multirobot formations is the leader-follower approach [94, 100, 204, 214] where one robot is selected as the leader and must be followed by the rest of robots. The task is usually implemented as only a single robot following some other autonomous agent. The follower must attempt to track the leader even though the leader undergoes possibly rapid random motion changes [105].

This approach can be extended to more than two robots. In that case, each robot references itself to one neighboring robot, using usually local information, maintaining a certain angle, ψ , and distance,

l, to it. Thus, the information needed is position and orientation of one robot close by and within line of sight. Keeping pre-set headings and distances from each follower to their leader robot it is possible to create and maintain a formation while robots are moving.

In this field, two works [79, 306] were developed at the Robotic Intelligence Lab of the Jaume I University. These works were useful as the basis for the work developed in this section.

Chiem [79] presented a simple method for the development of leader-follower formations of multiple autonomous robots. Each follower robot estimates the position and orientation of its leader with a color-tracking vision system, and builds a Bézier curve that describes the trajectory between its current position, and the position of the leader robot. This approach can be extended to more follower robots, while keeping a line formation. Also, Chiem stated that by the introduction of virtual points formations of different shapes can be generated, but it was not implemented.

Renaud [306] followed the same work as Chiem, that is robot formation control strategies based on a vision-based follow-the-leader scenario, but he concentrated on the reliability of the system. In that way, perception is enhanced by the control of a pan-tilt-zoom camera, which gives the follower robot a large field of view and improvement of the leader detection. Moreover, bidirectional and non-oblivious robot control is proposed, with the use of odometry to detect outliers in the follower vision-based pose estimation, and the leader path planning taking into account visibility constraint.

The application described in this section is a continuation of the previous ones but extended to multiple types of formations by using virtual points as explained in [79]. For this reason, some of the techniques and resources mentioned in the previous papers are used. An important value added to the previous works is the cooperation among the robots. The main idea is to make it feasible for a heterogeneous team of four robots, which is composed of one robot with a laser range-finder, one robot with a color camera and two plain robots without any sensor, to navigate through an environment in such a way that the robots with sensory power help the robots without it.

Before presenting the approach followed in this section, it is necessary to define some specific terms and specifications. As specified in [29, 131], after analyzing different types of geometric formations that multirobots can create, they agree that a geometric formation consists of three main parts: conductor, leader and followers. *Conductor* is the robot at the head of a group in a formation, and is the responsible of leading the group and all the other robots will follow it. *Leader* is the robot who takes the decisions about the formation. Finally *Follower* is any robot in a formation except the conductor, including the leader.

In any formation application, there are two phases, first creating the formation, and then maintaining the formation while the robots move. In the first phase, it is necessary to detect and localize the robots in the environment, for this end, the robot with the camera, the leader, searches in the environment the rest of robots and localize them with respect to itself by means of a colored target that the robots carry on. Once the robots are localized, it is easy to create the formation around the leader. Once the formation is created, the second phase is actived. In order to get the robots moving in a coordinated way, a *conductor-referenced system* is used, but using displacements of the conductor position, in the form of virtual points, in order to create the desired formation. That is, each robot will follow a virtual point, which is determined by the position of the conductor at any moment.

To determine the relative location of other robots, the leader uses the visual information obtained from the pan-tilt-zoom on-board camera. Camera images are used to detect other robots and to determine the relative position of the detected robot and its orientation with respect to the leader. Each robot carries a colored target that helps the leader to recognize it and calculate their position and orientation. Moreover, the zoom is used to enhance the perception and get a higher accuracy and a larger field of view.

Robot localization has been recognized as one of the fundamental problems in mobile robotics [129]. Using sensory information to localize the robot in its environment is the most fundamental problem that has to be solved in order to provide a mobile robot with autonomous capabilities [90]. Most of the existing work in localization is addressed to the localization of a single robot by itself. However, if robots can detect each other, there is the opportunity to do better. When a robot determines the location of another robot relative to its own, both robots can improve the accuracy with which they localize each other.

Vision has been widely used to get exteroceptive information in order to detect and localize robots. Although omnidirectional cameras have been used in the detection and localization of robots [94], directional cameras suppose a better option due to their much lower cost [321] and because they have complementary performances despite the visibility constraints [251]. Regarding the image processing, color has been widely used to achieve robot detection [93, 131, 251]. However, the robustness of color detection with respect to light conditions can be a major source of failures [92].

Also, the use of the zoom has been used in the context active vision [23] or visual servoing [164]. For using the zoom, it is necessary the explicit knowledge of intrinsic parameters from the calibration of the camera [80, 225].

The control of the maintenance of the formation while the robots are moving is performed using a decentralized process, where each follower decides which movements must be performed in order to follow the movements of the conductor. The conductor is specialized in navigation because it is using a laser range-finder to build maps and achieve the localization and navigation tasks. The actual position of the conductor while it is moving is sent to every follower. When receiving the position, each follower calculates the virtual point that it must follow. This virtual point, as mentioned previously, consists of a displacement in the position of the conductor that allows it to create different formations, not only the leader-follower line formation. Once the virtual point is calculated, the follower computes the trajectory it must follow in order to arrive from the current position to the estimated conductor relative position. For the calculation of this trajectory, Bezier curves are considered, as in [79, 306].

As it was seen in 2.3, other options can be used in order to maintain the formation, not only the conductor-referenced system. This system is very similar to the point-referenced system [199, 205, 262], where each follower determines its position in the formation in relation to the leader robot and it is responsible for the maintenance of the formation. The difference between the approach described in this system is that in this case the leader is not the same robot as the conductor and the leader itself is also a follower.

Another possibility is the unit-center-referenced formation [29, 203], where the average of the x and y positions of all the robots is calculated and used as a reference point for the robots so that they can calculate their own formation position. In this case, the perception by each robot of the other members produces a redundancy which increases the formation robustness.

Finally, in a neighbour-referenced approach, each robots maintains its position relative to one other predetermined robot, a neighbour [131, 251, 348]. This option can not be a good choice if failures are to be avoided, because, if a robot that works as reference fails, the follower robot can not detect this and so adjust their position or the positions of other components of the formation, so the formation will fail. If the failure can be detected, the most common robot attitude is the re-formation of the robots, but this can not be the best option if the failure is produced by a vision failure or a temporary occlusion.

The research presented in this section only concentrates on how to achieve the creation and the maintenance of a formation. But as it was seen in section 2.3, other problems are of special interest in the domain of multirobot formations. Regarding the types of formations that it is possible to form with the team of robots available, all the formations must have the conductor in front so that no robot can interfere with the laser range-finder sensor and thus produce bad readings. The rest of robots can not be arranged forward or besides this robot. With regard to the switching formation problem, it was not implemented, but with the definition of the formations in the systems and the method use to create the formations, it will be possible to switch the formation at any moment by simply stopping the formation and rearrange the robots in a new formation. Finally, in relation to the avoidance of obstacles, by means of the method used in the navigation and moving of the formation, all the static obstacles in the map are avoided, but no avoidance for dynamic obstacles is provided.

5.1.1 Design of the application

The objective of this applications, that is the creation of multirobot formations, is to demonstrate that the Acromovi architecture is suitable for the purpose for which it was created, to give the necessary support to the robot team for the programming and execution of cooperative tasks.

Formations can be defined as spatial structures made up of a set of robots where each one represents a point or vertex of a certain structure, and the imaginary lines between the vertexes or robots represent the structure contours. These formations are usually linear or polygonal in shape. But, when more than one robot is moving in a common space, each one of them is transformed into a mobile obstacle for the rest. For this reason a certain degree of coordination among the robots is necessary. The coordination consists of a movement synchronization so that each robot executes its tasks without producing a collision between them.

The main objective in this section is to develop a system able to create and maintain a formation while the robots are moving. It is important to remark that the available team for the development of the application is composed of robots with different capabilities. As common equipment, each robot has an onboard computer and can communicate with the rest of robots through a wireless network. But only one robot carries a laser range-finder to help it in the navigation and localization tasks, and only one robot carries a pan-tilt-zoom color camera to detect the rest of the robots and calculate their position. The rest of the robots are “plain” robots, that is, they do not have any type of sensor to help them in the resolution of the tasks assigned. It is only by cooperating with each other that these plain robots can perform the assigned tasks in a correct way, that is maintain the formation while moving.

As seen before, the applications consists of two phases, the first one, the creation of the formation, has as its objective getting the group of mobile robots, which are distributed in the environment, organized into a predefined formation among the ones that the group knows. This must be performed by themselves, without the help of any human operator. The second phase involves the movement and maintenance of the formation.

The first phase, the autonomous creation of the formation, was developed during a staying of the author of this thesis at the “Bristol Robotics Laboratory” in the University of the West of England, Bristol. The team of robots available at this laboratory consisted of three Linuxbots robots, see section 1.2.1, without any type of sensor mounted on them, and only one Pioneer-2 robot, see section 1.2.1, with a pan-tilt-zoom color camera, and all the robots had the capacity to communicate by means of a wireless network. The robots can be seen in figure 5.1.

As in this phase there is no intention to move the formation, no robot plays the role of conductor. The Pioneer-2 robot plays the role of leader and is the responsible for organizing the robots in the formation. The rest of the robots, the Linuxbots, play the role of followers, but in this phase they only obey the orders of the leader and no following action is performed.

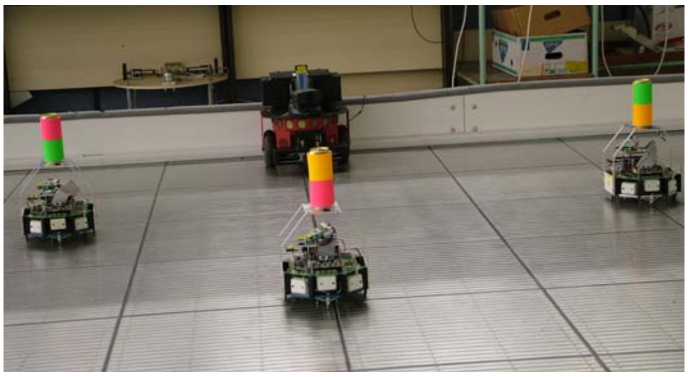


Figure 5.1: Team of robots used in the creation phase

The creation phase can be divided into two sub-phases, the first one consists of the detection and calculation of the position of the robots in the environment by the leader, using for this purpose the camera in order to detect the colored target. To this end, the leader first searches for a certain color pattern and then calculates the position of the robot that carries this pattern in relation to itself. Once all the robots are localized, in the second phase, the leader sends to all the robots the position that they must occupy in the formation, and the robots, knowing their actual position and the position where they must be, move to position themselves in the formation. The creation phase is repeated twice, so that final positions of the robots in the formation are more precise.

The second phase, the maintenance of the formation while moving, was developed at the "Robotic Intelligence Laboratory" in the Jaume I University in Castellon. Now, the team of robots is slightly different. All the robots of the team are Pioneers-2, but each one with different capabilities. Thus, there is still one robot with the camera, which also plays the role of leader. There is one robot which has a laser range-finder and plays the role of conductor, that is, it is in charge of driving the team by a previously defined path. Finally, there are also two plain robots, which are the *followers*, and their only mission is to follow the conductor and obey the orders of the leader. In this case, the leader also acts as follower, because it must maintain the formation in order to get the information for the plain robots, so it must follow the conductor as well. Moreover, all the robots have the capacity to communicate by means of a wireless network. This new team of robots can be seen in figure 5.2.

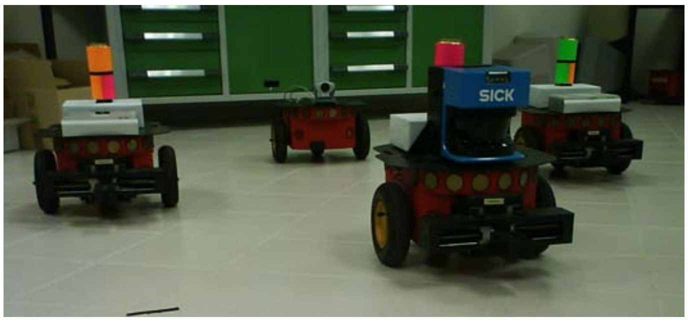


Figure 5.2: Team of robots used in the maintenance phase

This phase is actived once the formation is created. The conductor indicates the movement to be followed by the rest of robots in the formation and each follower decides which movements it must in order to follow the movements of the conductor. At each step, the actual position of the conductor is sent to every follower. The follower computes the trajectory to follow in order to get from its current position to the virtual point derived from the conductor position.

As it will be seen next, with this approach the robots are able to move in formation, but only in simulation environments, where the odometry error is controlled. When this approach is used in physical robots, the odometry errors make it useless. In order to improve the performance of the approach, a new agent is introduced in the system. In this case, the camera that is carried by the leader is used to reduce the odometry errors of the robots while they are following the conductor.

In this new approach, the types of formations that the system can create must be modified. Now, the leader must always be at back of the formation in order to get the rest of the robots in the visual field of the pan-tilt-zoom camera (180°). Also, occlusions in the formations must be avoided.

The leader can detect the position of the conductor and the rest of robots and at the same time that the formation is moving, it monitors the position of each robot and indicates to them the possible variations that each one must perform in its movement in order to maintain the formation. This is repeated until the robots arrive at the desired position.

5.2 Determination of feasible formations

In this section, all the formations that can be created for a team of four mobile robots are shown. Because of the limitations on the sensory input of the robots of the team, not all the possible formations which can be made with a group of four robots will be available.

The formations are conceptually defined by means of a matrix which indicates whether there should be a robot in a cell or not, and which robot should be there, identified by its ID, that is, for each formation, each robot has a specific position defined by its ID.

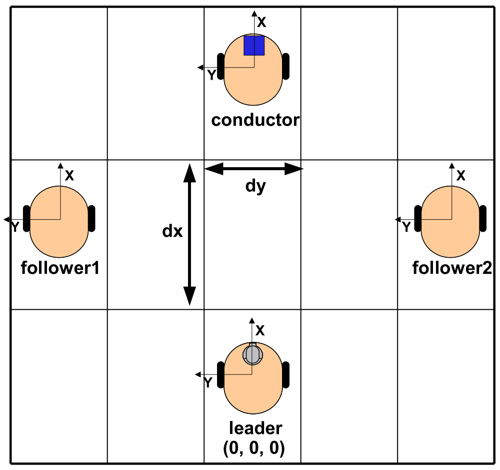


Figure 5.3: Example of matrix defining a formation

In these formations, the conductor must always be in front and no other robot can go in front or beside it. This is due to the fact that the conductor uses a laser range-finder with 180°of "visual field" in order to localize itself in the map and to follow a predefined path. If any robot is in the visual field of the laser, it can produce false readings and consequently, the localization of the conductor can be wrong and so it cannot follow the path. An example of a formation defined with a matrix can be seen in figure 5.3.

In the matrix two different sizes are defined, which correspond to the distances between the squares of the matrix in the vertical (dx) and horizontal (dy) ways. Thus, if the gap between the squares is 0.5 metres, and one robot is separated 2 squares from another robot, the separation between these two robots will be 1 metre. In that way, in figure 5.3, the formation will be a triangle, where the separation between the robots will be 1 metre.

With this vision, many different types of formations can be created, from formations with only two robots, the conductor and a follower, up to formations of four robots, the conductor and three followers. Some examples are depicted in the following figures. Although in the figures is not indicated which robot is the leader and carries the camera, it is easy to deduct that this robot must be one of the robots at back with sufficient visibility of all the other robots.

Formations with two robots

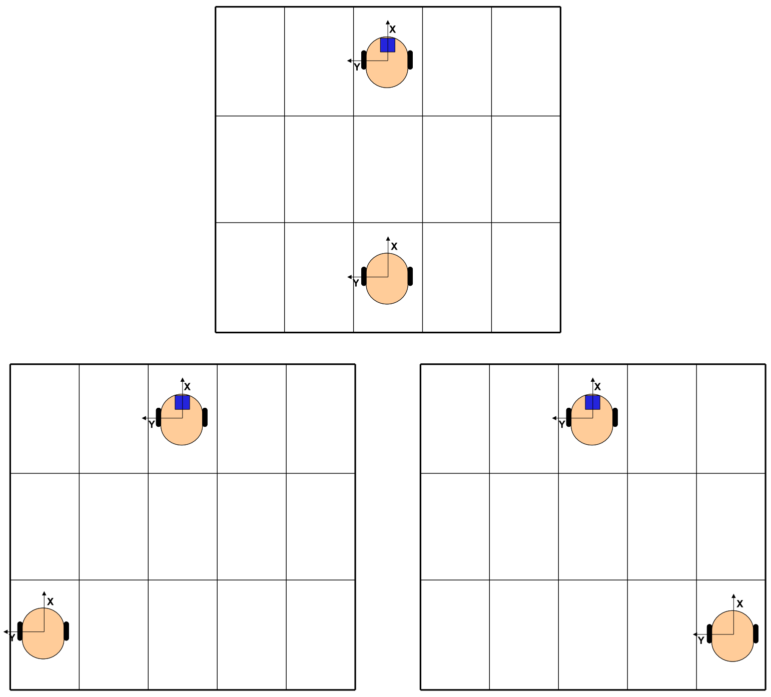


Figure 5.4: Examples of formations with only two robots, the conductor and one follower

Formations with three robots

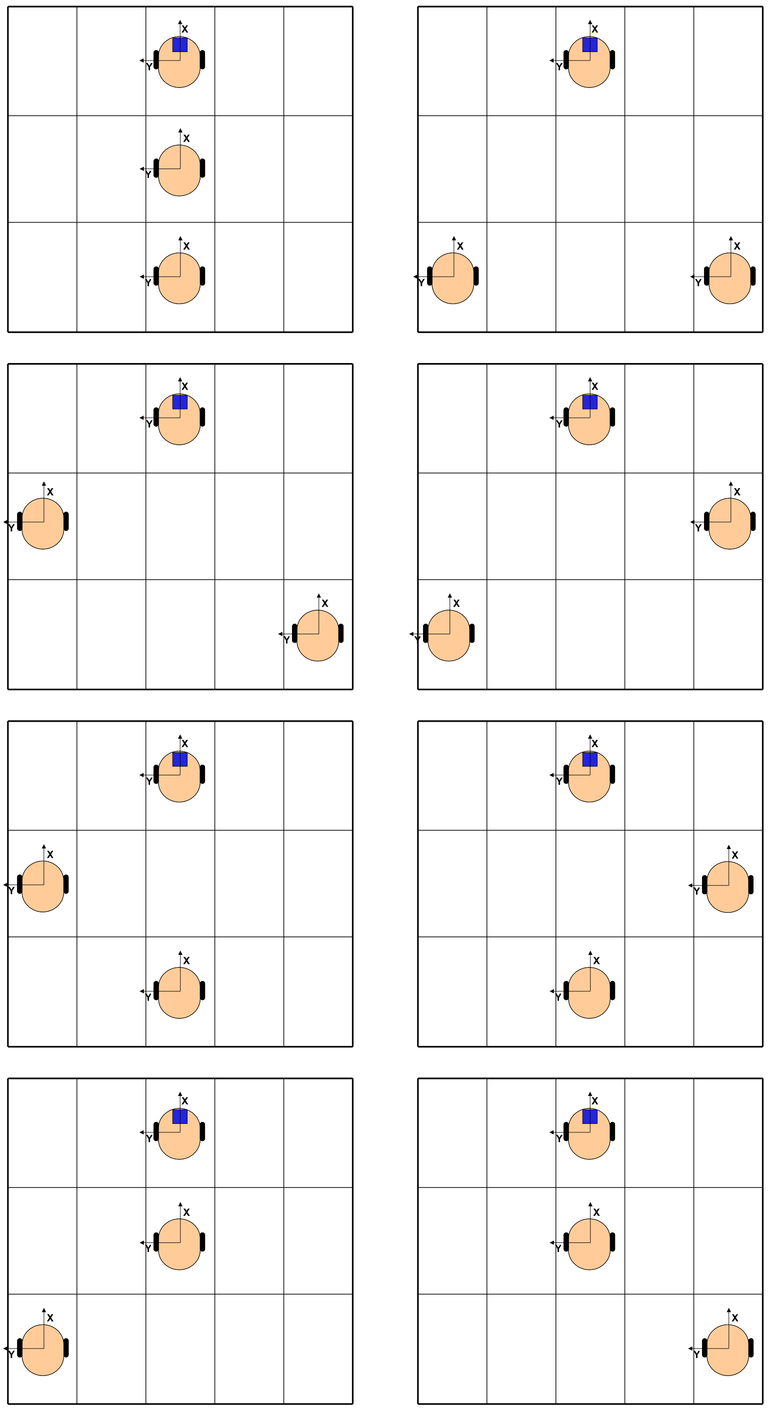


Figure 5.5: Examples of formations with three robots, the conductor and two followers (1/2)

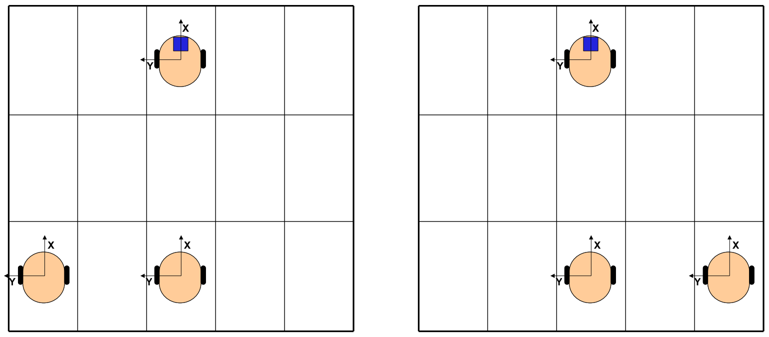


Figure 5.6: Examples of formations with three robots, the conductor and two followers (2/2)

Formations with four robots

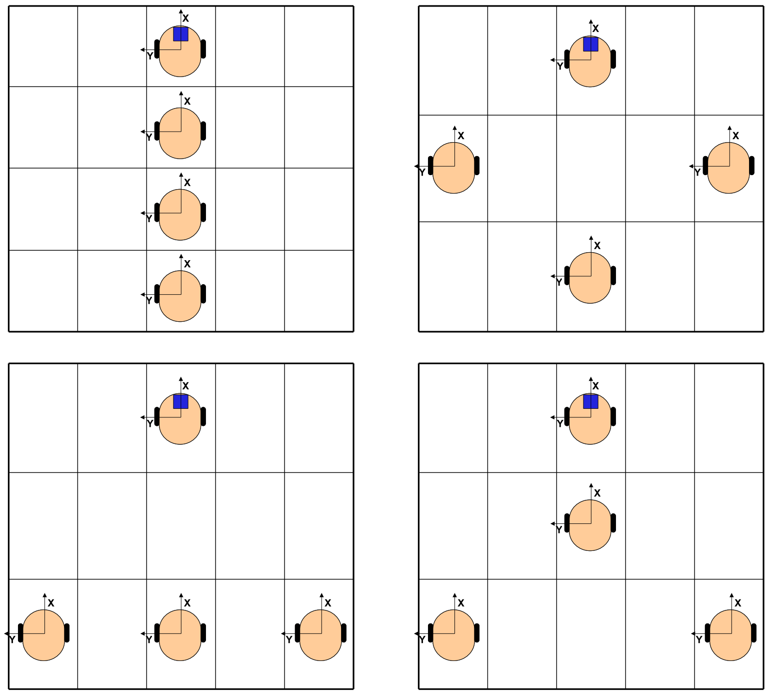


Figure 5.7: Examples of formations with four robots, the conductor and three followers (1/2)

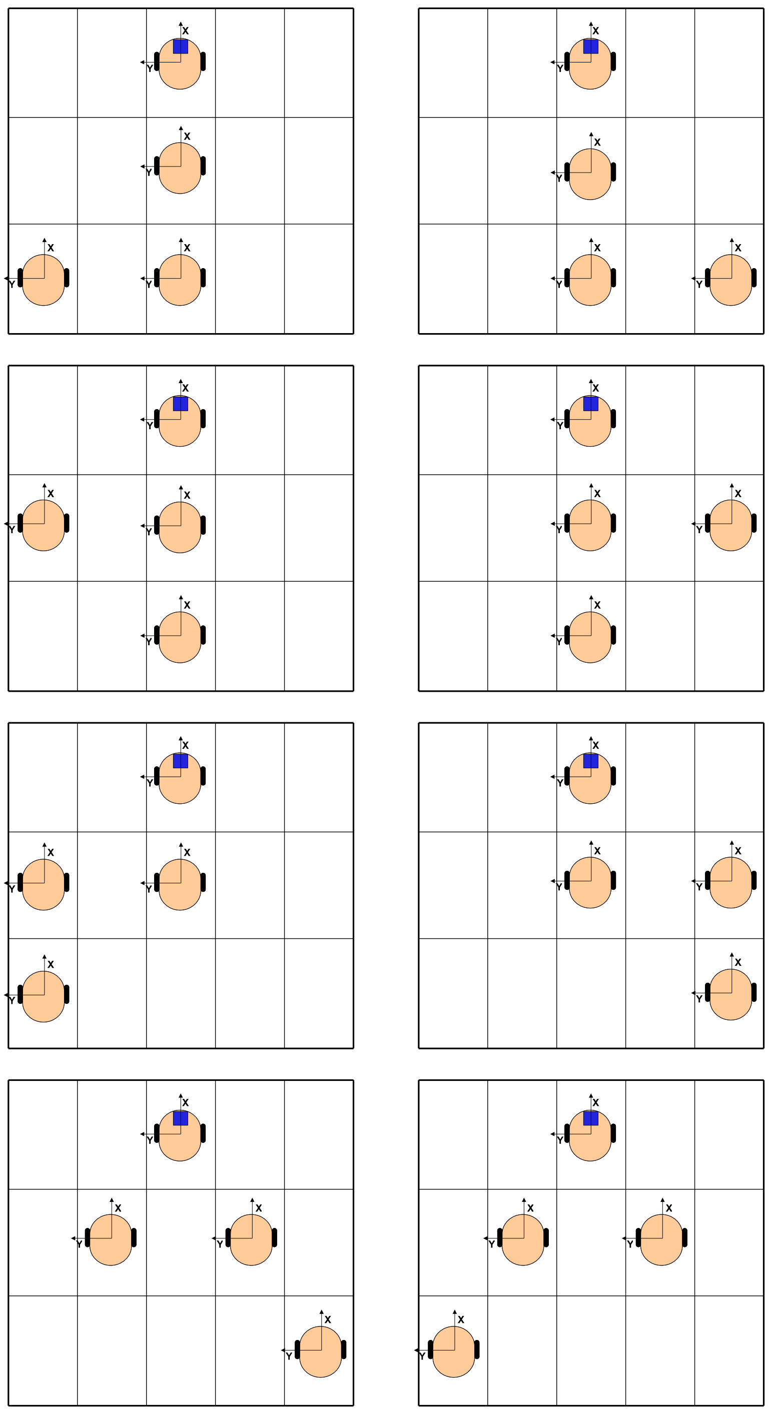


Figure 5.8: Examples of formations with four robots, the conductor and three followers (2/2)

5.3 Autonomous creation of the formation

This section describes how the formation can be created in an autonomous way. Having the robots distributed in an environment, they are able to organize themselves in a formation without the help of any human operator. To this end, the leader uses the camera that it carries to detect and localize the rest of the robots of the team and indicates to them which is the position that they have to occupy in order to create the formation. This process is explained in more detail in the following sub-sections.

5.3.1 Detection of the robots in the environment

The first step is the detection of the robots in the environment, in order for this decision to be taken, the leader uses its camera to detect a series of color patterns which identify each one of the robots in an unequivocal way. To detect the colors, the Mezzanine program is used.

Mezzanine [165] is an overhead 2D visual tracking package intended primarily for use as a mobile robot metrology system. It uses a camera to track objects marked with color-coded fiducials and infers the pose of these objects in world coordinates. Mezzanine works with most color cameras and can correct for the barrel distortion produced by wide-angle lenses.

Mezzanine consists of three main components or parts. Mezzanine itself is the tracking program, mezzcal is a tool for calibrating the tracking program, and libmezz is a library for communicating with mezzanine.

In order to use mezzanine for the purpose of the work explained here, it was necessary to modify the philosophy of the program. In this case, it is necessary to use the program not overhead but to get images from the front of the robots, so that, many of the features of Mezzanine, such as the inference of the pose of the fiducials, are not available.

Mezzanine is used only for the detection of the colors of the patterns that are used to recognize the robots. And with the information that is collected from the mezzanine system, it is possible to localize the robots in the environment and calculate their pose (position and orientation) with respect to the leader.

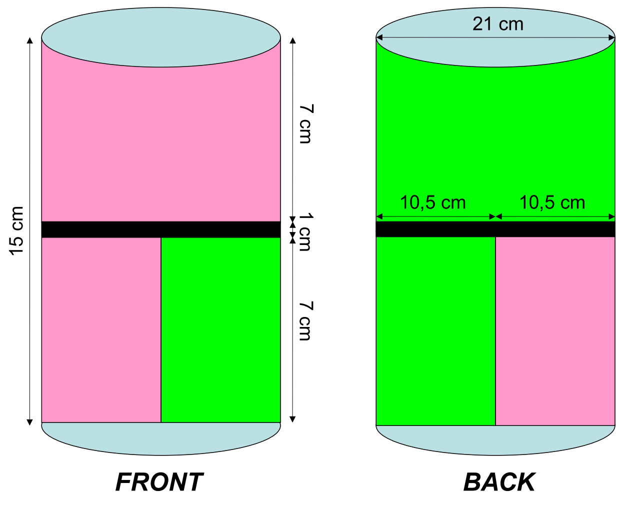


Figure 5.9: Dimensions and color layout of the IDs

The color pattern that the leader has to search for is created with a beer can of half a liter covered with colored cards in a specific layout, because each ID is unique. In figure 5.9, it is possible to see the sizes and dimensions of the target and the color layout at the front and at the back.

As it can be seen in the ID, there are two different parts separated by a black zone. These two parts are formed with the same colors and the same cards, but there is a 90°difference in orientation between the ID cards which means that thay are read as different cards. That is in this way to get two different readings of the orientation of the can and thus getting more accurate estimations.

All the three patterns used by the robots can be seen in figure 5.10.

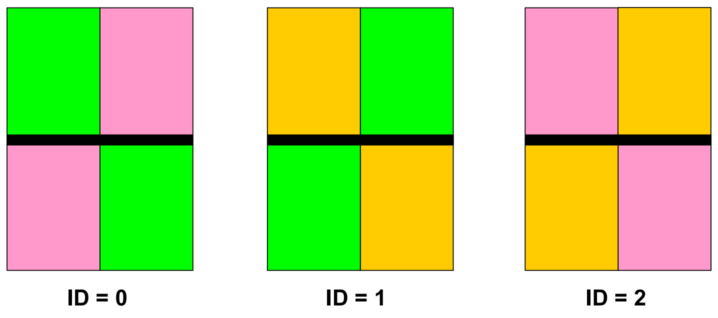


Figure 5.10: Different color patterns used to identify the robots

Since each color card corresponds to the ID of a different robot, the leader is able to recognise each one. Also, with this pattern it is possible to calculate the pose of the robot in relation to the leader. It is easy to recognize which robot the camera is seeing, it is simply necessary to pay attention to the layout of the colors. Mezzanine can detect several colors at the same time and group the different areas of the same color in blobs. With the information associated with these blobs it is possible to know which ID the camera is seeing at that moment.

The movement that the camera performs to find these IDs is firstly horizontal movement of 180 degrees from left to right. If nothing is found in this movement, the camera increases its vertical position in 5 degrees up to a maximum of 30 degrees. If when this process has finished, still any robot has not been identified, the leader executes a 180 degrees turn and repeats the same process until all the robots are found. In this way, the leader searches all the space around it for the other robots.

Throughout the searching process, mezzanine is monitoring all their channels where it has assigned a predefined color, and in the moment that it finds anyone of them, the camera is stopped. From this moment, a centering process begins. This new process tries to center the pattern found in the middle of the image. To this end, because the robots are in movement, it is important to center the target in a minimum number of movements, and at the same time it is important to maximize the zoom of the camera to make a better identification in the following phases.

In order to center the target, when mezzanine detects one blob of any color bigger than a certain size, the robot stops the searching process previously described. The size has to be big enough to rule out possible errors of the program or reflections of the target. From this blob it is possible to know the position of its mass center in the image system, so the space between this and the center of the image can be calculated.

To translate this distance into a movement of the camera, it is necessary to know some intrinsic parameters of the camera, such as the focal length. These parameters can be obtained with a previous calibration of the camera. Having a calibration target, from the pinhole camera model of a camera, the relationships among the parameters of the camera and the parameters of the scene can be seen in figure 5.11.

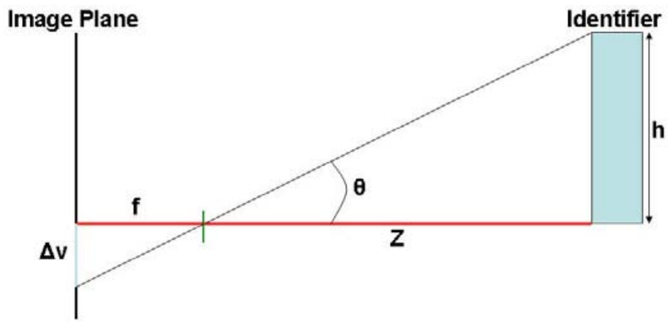


Figure 5.11: Relation among the parameters of the camera and the scene in the pinhole camera model

Knowing that (h) is the real height of the calibration target, (∇h) the height in pixels of the calibration target in the image, (Z) the distance from the camera to the calibration target, and (f_v) the focal length of the camera, it is possible to calculate the the focal length:

$$f_v = \Delta v \frac{Z}{h} \tag{5.1}$$

As the camera includes a zoom, the focal length must be calculated for each of these values of the zoom. After several tests performed in the laboratory by Vincent Robin during a stay there, he managed to model the behaviour of the focal length depending on the zoom. The function that models these behaviours can be defined as:

$$f_v = (0.0368323 - 0.0000128323 * z)^{-2} * 1/2 \tag{5.2}$$

being z the value of the zoom that is desired. This function can be visualized in figure 5.12, and as it can be seen, the focal length does not follow a linear progression with the progression of the zoom.

Knowing the value of the focal length for the actual zoom of the camera, it is simple to calculate the movements that the camera must perform in order to center the blob in the image. There are two movements that have to be made, in the pan (∇p), that is, in horizontal, and in the tilt or vertical (∇t). These values can be calculated as:

$$\begin{array}{c} \nabla p = \arctan\left(\frac{(x-x_i)}{f}\right) \\ \nabla t = -\arctan\left(\frac{(y-y_i)}{f}\right) \end{array} \tag{5.3}$$

where x and y correspond to the coordinates of the mass center of the blob in the image, and x_i and y_i correspond to the center of the image. With these values the first blob that the robot finds can be centered.

Once the first blob of the target is centered, it is possible to calculate the optimal value of the zoom in order to reduce the detection failures of the targets of the robots and making sure of a better

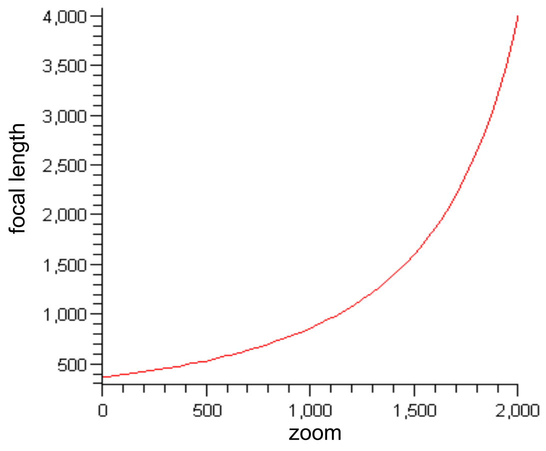


Figure 5.12: Relation among the necessary parameters to calculate the distance to the ID

approximation in the calculation of the position of the robot. These calculations are based on the previous work of Pierre Renaud [306] during a stay in this laboratory.

Prior to calculating the optimal zoom, it is necessary to calculate the optimal focal length, and with this value is possible to calculate the optimal zoom. In order to calculate the optimal focal length, as can be seen in the relations in figure 5.11, it is necessary to know the actual distance (Z) to the target.

$$Z = f_v * \frac{h}{\Delta v} \tag{5.4}$$

With this distance, and knowing which is the desired height (h_{des}) of the blob that the program needs to get the optimal configuration, and knowing the height of the blob in the image, the optical focal length can be calculated.

$$f_{op} = h_{des} * \frac{Z}{h} \tag{5.5}$$

Finally, as deduced by Renaud in his work, the optimal zoom can be calculated merely by knowing the optimal focal length.

$$z_{op} = 77928.35 * \left(3.6833e^{-2} - (2 * f_{op})^{-0.5}\right) \tag{5.6}$$

Once this calculation is made, the zoom is applied to the camera, and as the rest of the blobs or colors of the target are now visible, it is possible to identify the robot. This process is simple and merely perceiving the distribution of the different colors in the target, the different robots can be identified.

Next, in order to make the calculation of the position more precise, a new centering process is carried out, but this time taking into account the blobs of the other colors present in the image. The biggest blob of the other color in the image provides the system with enough information to center the target in the image. The new equations to calculate the movements of the camera to center the target are:

$$\begin{aligned} \nabla p &= \arctan\left(\frac{((x_1+x_2/2)-x_i)}{f}\right) \\ \nabla t &= \arctan\left(\frac{((y_1+y_2/2)-y_i)}{f}\right) \end{aligned} \tag{5.7}$$

Once the ID is centered on the image and with the maximum size possible, its position and orientation with respect to the camera, or the leader, can be calculated. To know its pose $(x, y, angle)$, it is necessary to perform some calculations with the image.

In order to calculate (x, y), the system needs to know the distance and the angle of the ID with respect to the leader, and from these values, calculates the position.

To calculate the distance from the ID to the image, it is necessary firstly to know some parameters. These parameters are the real height of the ID (h), the height in pixels of the ID in the image (∇h), and the focal length of the camera (f_v). With these values, the distance to the ID (Z) can be calculated as:

$$Z = f_v \frac{h}{\Delta v} \tag{5.8}$$

As can be infered from figure 5.11, the height of the ID is fixed, and the height in pixels of the ID in the image can be obtained from the blob information from mezzanine. The focal length of the camera can be obtained as explained before, merely by knowing which is the actual value for the zoom of the camera.

The precision of the approximated distance depends on the capacity of the system to recognize the specific colors of the cylinders, which is influenced by the prevailing lighting conditions. When the cylinder is lit from the side, their colors are preceived no longer uniform, making only part of the width of the cylinders visible to the leader. For an optimal approximation, good uniform lighting is necessary.

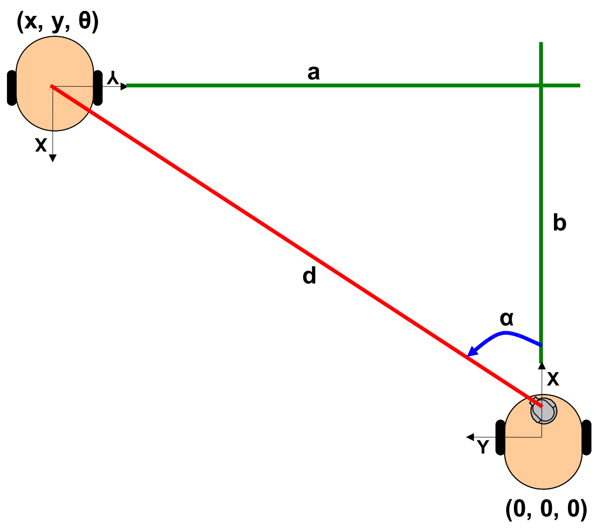


Figure 5.13: Relation among orientation and distance with the calculation of the ID position

The calculation of the orientation (α) at which the robot is depending on the leader is easier, since the agent in charge of the camera operation (CameraAgent) indicates the orientation that it has (α) at that moment. In figure 5.13, the relation between orientation and distance for the calculation of the ID position (x, y) can be observed.

When the distance and the orientation have been calculated, it is possible to calculate the position (x, y) as:

$$\begin{array}{c} d = \sqrt{a^2 + b^2} \\ \\ \alpha = \arctan b/a \end{array} \quad \Longrightarrow \quad \begin{array}{c} x = b = d \cdot \sin \alpha \\ \\ y = a = d \cdot \cos \alpha \end{array} \tag{5.9}$$

Once the position has been calculated, it only remains to calculate the orientation of the robot with respect to the leader. For this calculation, the two horizontal parts or colors of the ID are used, or rather, the relation between these two parts. In figure 5.14, some specific positions of the ID with the relation between the colored parts and which angle is assigned for those positions can be observed.

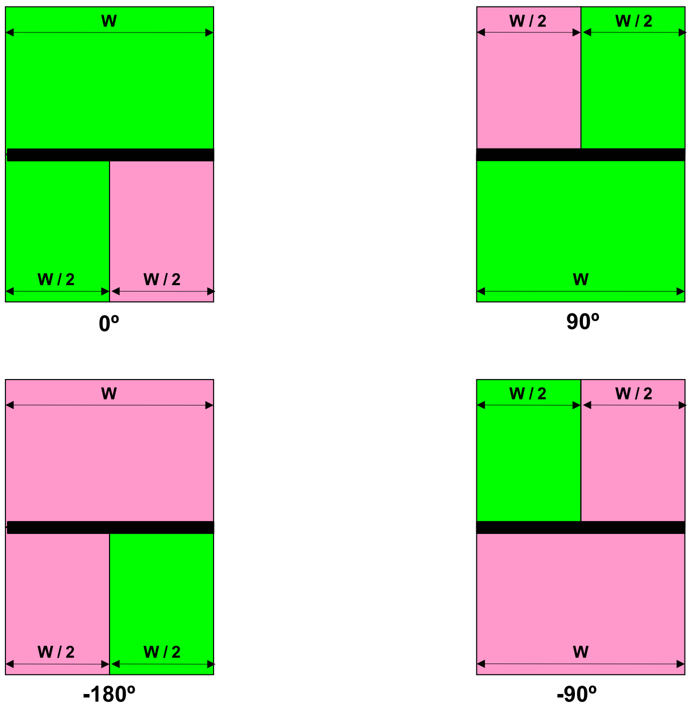


Figure 5.14: Angles assigned to the specific positions of the ID

As it can be seen, based on the existing relation between the horizontal sizes of the two colored parts, the orientation of the robot can be calculated. It can be observed that the upper layer and the lower layer have the same colors, but the lower layer has them with a specific turn in relation with the configuration of the upper layer. This is done to have two differentiated parts and to calculate the orientation of the upper and lower layer separately, and thus, making the calculation more precise. Moreover, as it will be seen below, in that way it is possible to avoid some positions that are not accurate enough in the calculation of the orientation.

In the calculation of the orientation using the target selected, it is very important to take into account the order of the colors. Regarding this, from the 0°position of the can to 180°position, the pink color is in the upper left position and grows until covers the complete side of the can. If the green color is in the upper left, the orientation will be from -180°to 0°, depending on the portion of the can occupied by this color. In the two cases, it corresponds the 90°or -90°when the two colors occupy the same portion of the can, as it can be seen in figure 5.14, but depending on the color in the upper left side, the orientation will be positive or negative.

From the relation of the left part (X) and right part (Y), the orientation of the robot can be calculated. The relation among the left and right parts and the entire width of the cylinder can be seen in figure 5.15.

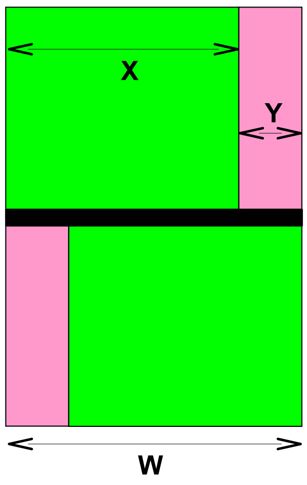


Figure 5.15: Relation among the parts to calculate the orientation of the robot

From the figure, it can be deducted when the left and right parts are equal that

$$\frac{2X}{W} = \frac{2X}{X+Y} \tag{5.10}$$

The behaviour of the cylinder when it is turning can be modeled as $\sin(\alpha - 90) + 1$. Joining this with the previous equation,

$$\frac{2X}{W} = \sin(\alpha - 90) + 1 \tag{5.11}$$

from which it can be infered the value of α as,

$$\alpha = 90 + \arcsin(\frac{2X}{W} - 1) \tag{5.12}$$

In the case of the negative orientation, the equation is similar,

$$\alpha = -90 + \arcsin(\frac{2X}{W} - 1) \tag{5.13}$$

In figure 5.16, it can be seen the graphics in radians that model these to functions.

Moreover, it is necessary to take into account the four exceptions to the general rule seen previously in figure 5.14. If there is only one color in one of the parts of the ID (upper or lower), then there

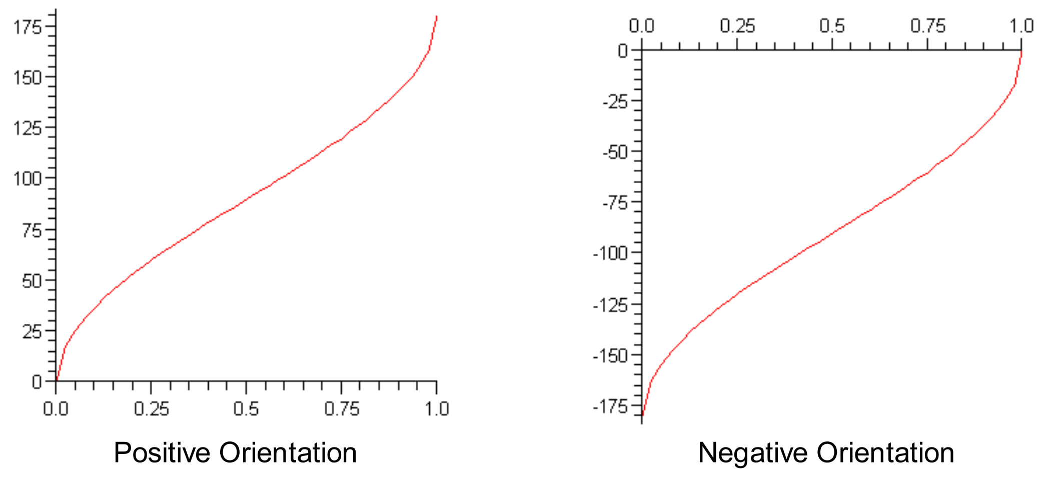


Figure 5.16: Functions that model the orientation of the robot

is one of these special cases. Simply, distinguishing the order of the colors in the remaining layer, it is possible to recognize the spacial case in question. The degrees associated with each special case can be seen in the figure.

Also, as it can be seen in the graphs in figure 5.16, there are two zones in each graph in which the value of the orientation changes too fast and it may cause the estimation to be less accurate than desired. These zones are at the extremes with a width of 30°. When the calculation in one of the layers returns a value within one of these zones, the values that are taken for the orientation of the robot is the value that returns the calculation in the other layer. Due to the fact that the layers have different turns, the values that appeared in one layer when the other is in one of these situations are correct.

This calculation of the orientation of the robot can be easily extended to any of the IDs seen before in figure 5.10. It is only necessary to take into account which is the pattern of the colors. Furthermore, this identification system can be extended to any number of different IDs, the only limitation is the number of different colors that mezzanine is able to detect.

Once the robot's orientation is calculated, all the values necessary for determining its pose $(x, y, angle)$ in the environment with respect to the leader are available. This pose is sent to the corresponding robot so that it knows its position.

5.3.2 Creation of the formation

When the pose of the robots has been calculated, the next step is the creation of the formation. The leader decides which formation must be created, and where each robot must be positioned in order to create that formation.

The information about the position in the formation that each robot must occupy is sent to it by the leader, and in the moment of receiving that information, the robot calculates the direction it must take in order to arrive at the corresponding position and then moves in that direction. In figure 5.17, the initial and final positions of an example of formation can be seen.

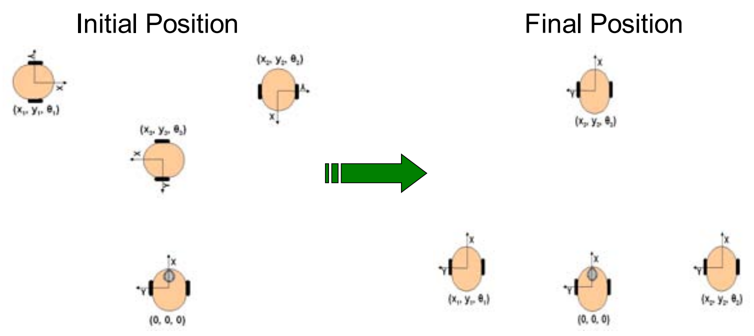


Figure 5.17: Example of the initial and final positions in the creation of a formation

Based on the matrix seen in figure 5.3, the positions which the robots must take up in order to create the formation can be extracted and sent to each robot. As it has been explained, the position which each robot must occupy in the formation has been indicated by means of its ID.

Once the robot has been informed of the position it must adopt, it calculates the necessary movements it must make in order to arrive at the desired position. The robot must carry out a turn in order to be pointing in the right direction to move to the final point (θ); next, the robot must move in straight line the distance corresponding to the difference between the two points (d); and finally, the robot must effectuate another turn to orientate itself towards the angle indicated in the final position (ω).

In figure 5.18 all the necessary movements to drive the robot from one position to another one can be seen.

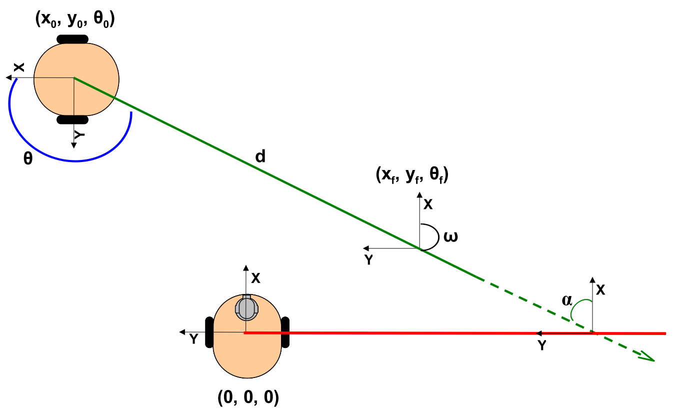


Figure 5.18: Movements to drive the robot from one position to another one

The formulas used to calculate the turning angles and the distance to move can be extracted from the geometrical relations shown in the figure. As can be noticed in the figure, the angle α is that which determines the direction that the robot must follow in order to get to the desired position.

$$
\begin{array}{ccc}
 & d = \sqrt{(x_f - x_0)^2 + (y_f - y_0)^2} & \\
\tan\alpha = m = \left(\frac{y_f - y_0}{x_f - x_0}\right) & \Longrightarrow & \alpha = atan2\left(\frac{y_f - y_0}{x_f - x_0}\right) \\
\theta = (\alpha + \pi) - \theta_0 & & \omega = \theta_f - (\alpha + \pi)
\end{array}
\tag{5.14}
$$

5.3.3 Implementation of the application

In order to introduce this application in the Acromovi architecture, five new agents have been implemented to give support to all the parts explained in this section. A brief description of each agent is provided.

TrackerAgent is the agent in charge of managing the search for the robots, and it is this agent that receives the current positions of the rest of robots and sends to each one of the other robots the position to which they are supposed to move as well as their current position.

In order to perform the search for the target that the robots carry, there is the *SearchAgent*. This agent communicates with the agents of the camera and mezzanine to get information about the scene to look for certain blobs that can reveal a target.

When a target is discovered, the *CenterAgent* is the responsible for centering that target in the image and to maximize the zoom as much as it is possible to get the maximum of accuracy possible.

IdentifyAgent uses the information in the image, once it is centered, to identify the robot by means of the target, and to calculate its position in relation to the leader.

These four agents are executed by the leader, in each of the other robots there is an agent called *RobotAgent* which communicates with TrackerAgent to get the information concerning the current position of the robot and the desired position of the robot in the formation. It is also the responsible for moving the robot to the desired position in order to create the formation.

The relation among these agents and the messages that they exchange can be seen in figure 5.19.

As can be seen in the previous figure, TrackerAgent is the center of the whole process. It is in charge of sending the corresponding message to SearchAgent to initiate the search for the rest of the robots ("search_initial"). When SearchAgent detects a blob that can belong to any target it sends the information about the color of this blob to CenterAgent ("CH"). CenterAgent centers the target in the image and maximizes the zoom, and it also detects the other color present in the target. The information about the two colors ("CHX + CHY") and the all the blobs which form the target are sent to IdentifyAgent. This agent, with the information received, identifies the robot and calculates its position in relation to the leader. When this process has finished, IdentifyAgent sends the information about the actual position of the robot and its ID to TrackerAgent. When TrackerAgent receives this information, it sends another message to SearchAgent ("search_initial") in order to find more robots in the case that not all of them have yet been detected, or sends the current and desired positions to each robot so that they move to create the formation.

Each robot has a RobotAgent instance running which is waiting for the messages from TrackerAgent. When this agent receives the actual and desired positions from TrackerAgent, it calculates the necessary movements it must make in order to take up the desired position. Once the robot has arrived to the desired position, it sends a message to TrackerAgent reporting this fact.

When TrackerAgent has received the messages from all the robots confirming that all the robots are in the desired formations, it begins a new round of localization in order to get more accurate

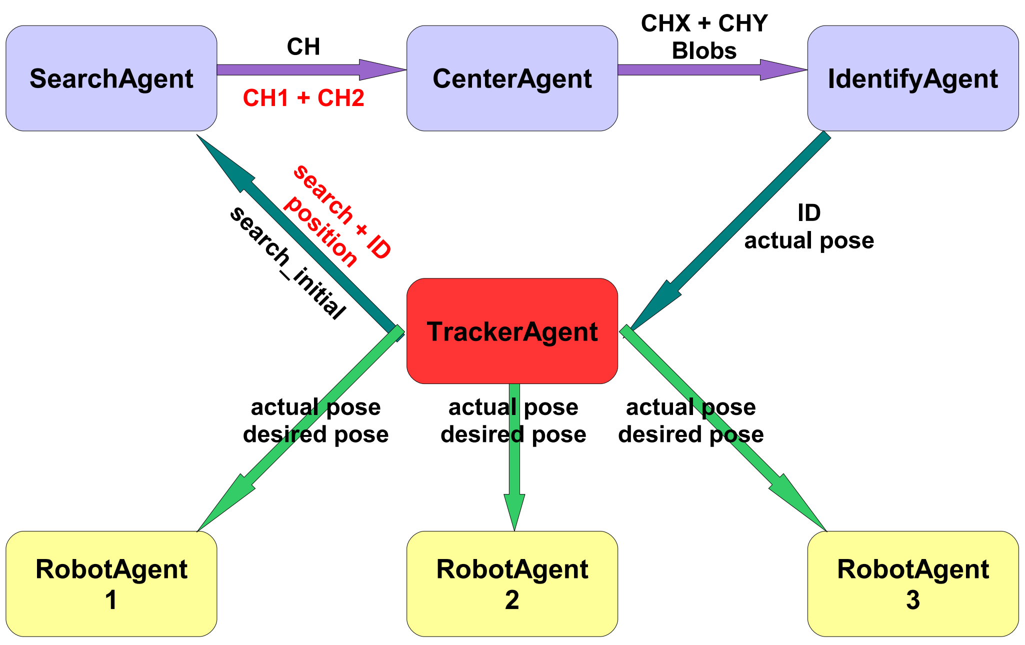


Figure 5.19: Agents and interaction among them to detect and localize the robots of the team

results. This time, instead of sending Searchagent the order to search for the robots in the environment, it sends the order but includes the ID of the robot to be searched for and the position where it should be ("search + ID" + position) to make the searching process shorter. SearchAgent in this case performs a limited search in a -30°to 30°interval around the position indicated. When it finds the target, it sends to CenterAgent the information about the two colors presented in the target ("CH1 + CH2"). CenterAgent performs the same process as in the previous phase and sends to IdentifyAgent the new values. IdentifyAgent calculates the new position for the robot and sends this information to TrackerAgent.

Finally, TrackerAgent sends to the robots the new positions. When each robot receives the new position, it moves again in order to reach the final position, but only if the error in the previous position is bigger than a predefined threshold. When the robots finish their new positioning process, they indicate this fact to the TrackerAgent, and in this moment the formation can start its movement.

5.3.4 Results in the creation of the formation

From the experiments performed in order to evaluate the precision in the creation of the formation, it can be concluded that the robots perform the expected behaviour. The leader is able to detect the rest of the robots by means of the camera and can calculate their position, and the robots are able to move from their position to the the desired position in the formation.

In the images in figure 5.20, the development of one of these experiments can be observed, and it can be seen how the robots are able to organize themselves in a certain formation independently of their initial position.

But the results are not quite exact. Due to errors in the precision of the camera and errors in

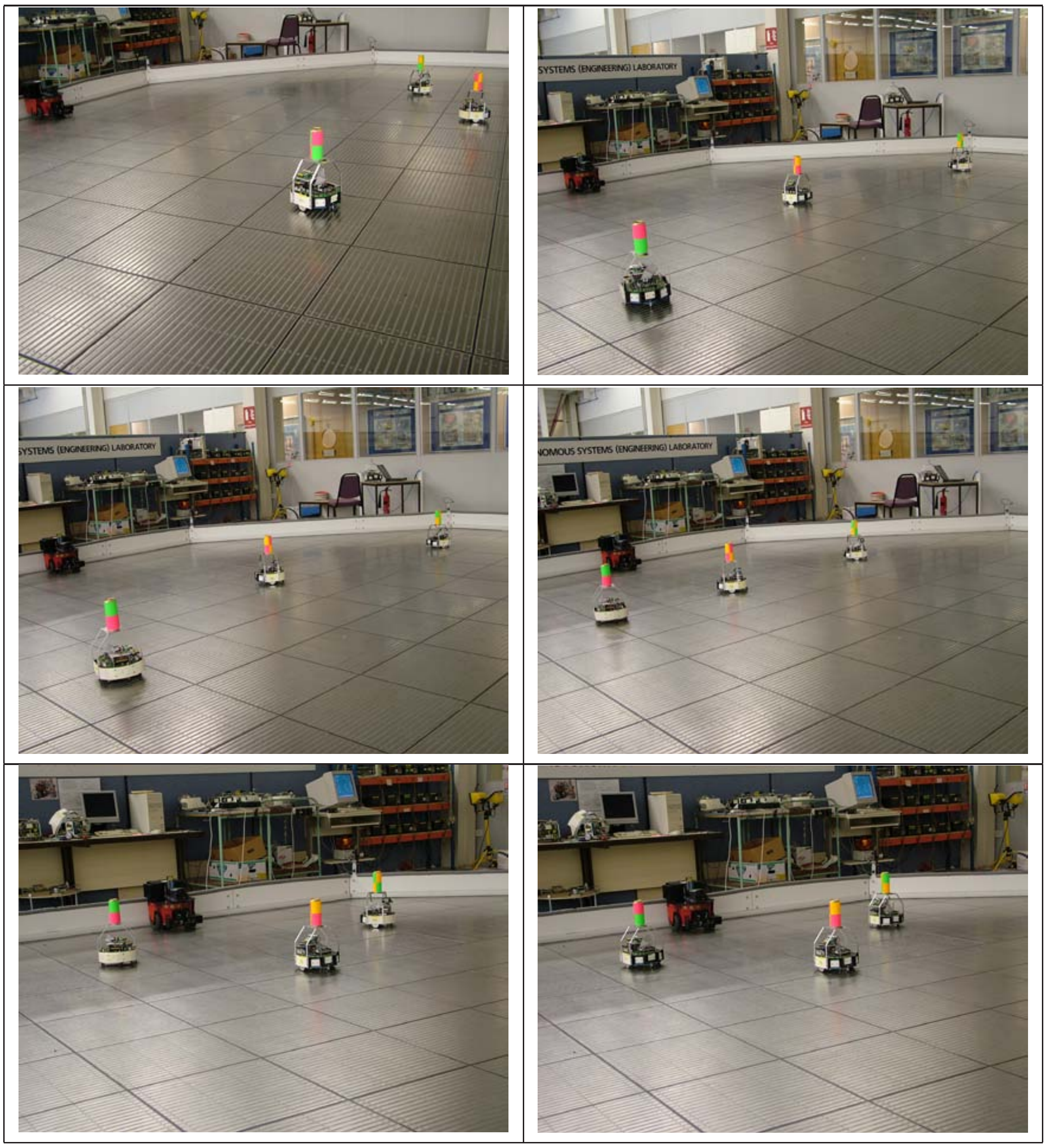

SYSTEMS (ENGINEERING) LABORATORY
SYSTEMS (ENGINEERING) LABORATORY
NOMOUS SYSTEMS (ENGINEERING) LABORATORY

Figure 5.20: Example of the creation of a formation

the odometry of the robots, at first the robots did not reach the correct position. That is, a small discrepancy was detected between the desired position of the robot and its current final position. This error, after several tests and measurements, could be estimated from 10 to 20 centimetres when the robots had moved from a distance of 6 or 7 metres, a not unreasonably margin of error.

With the intention of reducing that error, when the robots get to the final position, another phase for recognizing the robots and their positions is performed, and the robots move again until the final position of the formation, if it is necessary. With this second recognizing phase, the error decreases to just a few centimetres. This margin of error is considered quite satisfactory.

5.4 Maintenance of the formation

In this section it is described how a formation can be maintained while the robots are moving in the environment. The formation is composed of a maximum of four robots, where one of them, the conductor, is equipped with a laser range-finder that allows it to navigate in buildings and follow a path determined from the map of the building, the goal point, and the obstacles.

In the formation, the position of the robot followers can be controlled by the position of the conductor if all the robots are arranged in a line formation. In other cases, when the followers are positioned to the right or left of the conductor, virtual points are added to the system, as explained in [79]. These virtual points are calculated by applying a displacement in the conductor position to the left or to the right, depending on the desired formation. The followers, in this case, instead of following the conductor must follow these virtual points. This arrangement is shown in figure 5.21.

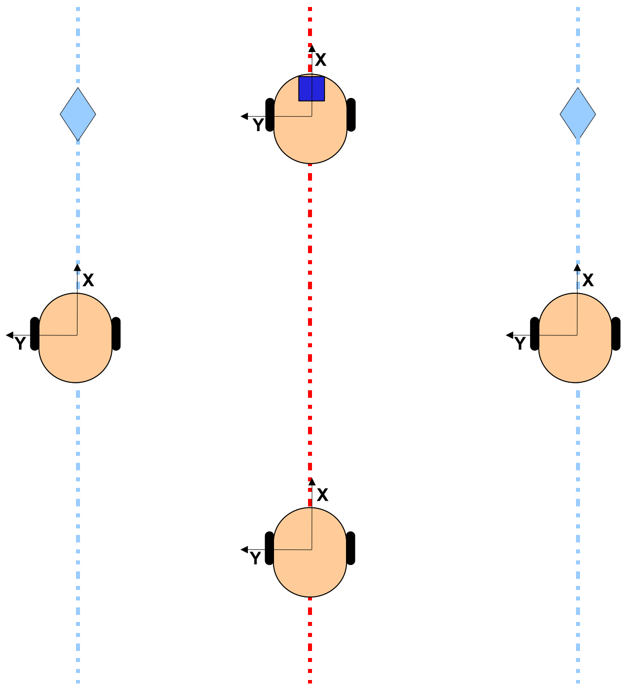


Figure 5.21: Formation using virtual points as reference for followers

Different possibilities have been tested for this application. Firstly, it was desired that the robots, only using the information from the current position of the conductor, and consequently the virtual points, could maintain the formation based on their odometry. As it will be seen in this section, this approach worked appropriately in simulation experiments, but when it was tested in real robots, this approach failed due to the odometry errors in the robots.

To solve this problem, the decision was taken to try to use the camera of the leader robot to improve the navigation of the robots and reduce the odometry errors. Monitoring the position of the followers while the formation is moving, it is possible to indicate to them when they have lost the position in the formation and which is their current position. The followers, in this case, must perform a modification of their trajectory in order to put themselves once again in the correct position in the formation.

5.4.1 Improvement of the conductor navigation

As it is necessary for the robots to move in order to check if they are able to maintain the formation, the use of a robot with a laser system is proposed. This robot, by means of the laser, can create a map of the environment and with that map it is able to localize itself in that map. It is also able to follow a predefined path in order to get to a certain position.

The first option for implementing the localization and navigation was using the Saphira software which is included in the Acromovi architecture, as seen in chapter 3. But this software was out of use and ActivMedia, the construction company of the robots, did not provide support for it. A new software had been introduced by this company to solve the problems of localization and navigation in the robots, the Advanced Robotics Navigation and Localization (ARNL) [310].

ARNL is a software development library for localizing and navigating mobile robots within mapped indoor environments. ARNL uses the data obtained from a laser range-finder to localize the robot in the environment and to detect obstacles. It also uses the laser data together with robot odometry and a map in order to navigate through the environment by planning and then following a certain path. One of the main characteristics of ARNL is that all the tasks are performed in background while the main program continues its execution.

ARNL is divided into two different tasks, the localization and the navigation tasks. It is important to note that the localization task can work by itself without the navigation task being active, but the navigation task needs the localization task to work. The localization task is in charge of keeping the robot localized in the map using the laser range-finder and the robot's odometry data. In order to localize the robot in the map, ARNL uses the Monte Carlo localization algorithm. The navigation task is responsible for the navigation of the robot. It is possible to use this task to send the robot autonomously and safetely to a goal location in the map. To calculate the path to follow, ARNL uses the grid-based search, which computes an optimal path from the current position of the robot to a predefined goal avoiding all the objects indicated by the map. In order to follow this path, ARNL uses the dynamic window approach.

Separate libraries are provided by ARNL for the different tasks mentioned before. So it is easy to create applications that use only one or both libraries, depending on the purpose of the application. However, one disadvantage of ARNL is that it does not perform global localization, that is, the robot must be placed in a known position so that it will be able to localize itself in the map.

In order to introduce this new software in the Acromovi architecture, and thus make it available for the cooperative tasks that this architecture can implement, new libraries and agents have been implemented following the process explained in section 3. In figure 5.22, the new agents and libraries created for improving the system with the facilities of localization and navigation are shown.

As can be seen in that figure, two new native libraries have been created to manage the access to the localization and navigation systems. This libraries communicate directly with the native software of ARNL and also provide some facilities in order to simplify the use of these libraries. As it can be observed, LocalizationNative and NavigationNative communicate between themselves. This is due to the fact that for using the Navigation task of ARNL, it is necessary to use the Localization task at the same time. In the immediate upper layer, there are the Java libraries that communicates with the native libraries by means of JNI. Finally, the last layer corresponds to the agents that use the Java libraries, and give support to the applications.

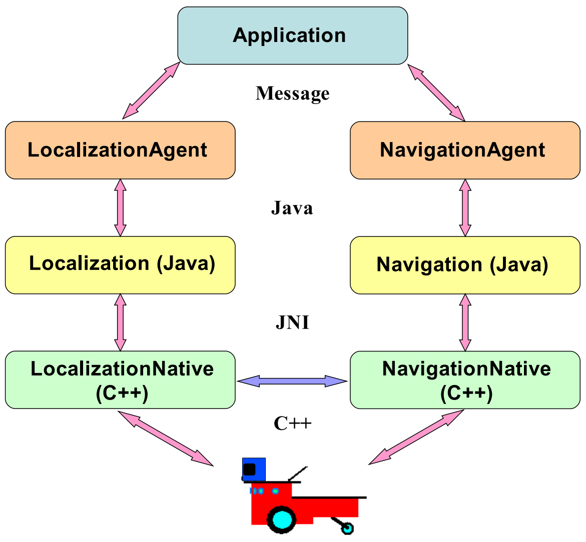


Figure 5.22: New agents and libraries implemented to integrate ARNL in the system

LocalizationAgent and Localization libraries, facilitate the initialization of the laser system and localization task, and loads the map of the environment and the necessary parameters for the localization. It also provides functions to consult the state of the localization, which is the actual position or which is the home position.

NavigationAgent and Navigation libraries, also facilitate the initialization of the navigation task and load the necessary parameters for it. It is also possible to indicate a certain position or goal and the navigation task will move the robot so that it follows a previously calculated path. Moreover, it offers functions to consult the state of the navigation, which is the current goal, the progress of the navigation, and the time or distance to arrive at the goal. Finally, two important functions that are provided by this agent are one function to cancel the navigation and one function to get the path planned for the navigation of the robot.

An example of a navigation task performed by the robot can be seen in figure 5.23. In this figure, the path planned by ARNL is indicated with a red line, and the path followed by the robot with a green line.

But the use of the navigation provided by ARNL has several problems. First of all the robot is controlled by the ARNL navigation task, and no parameter, for example lineal or angular velocities, can be modified to control the navigation of the conductor and make it smoother so that the followers can maintain the formation. Another problem detected after a few executions is that the robot does not follow exactly the trajectory planned previously, producing sudden movements of the robot, especially while turning.

After detecting these problem, the decision taken was to implement a new navigation task, but using the path planned by the ARNL navigation task. In this way, the application will communicate with NavigationAgent to ensure that this agent calculates the path, but once this path is calculated, the navigation is canceled and the path is passed to the application to use it to perform its own navigation. The path produced by the ARNL navigation task consists of a list of positions which are used by the robot so that it can move to its destination. This list of positions is used by the new navigation task

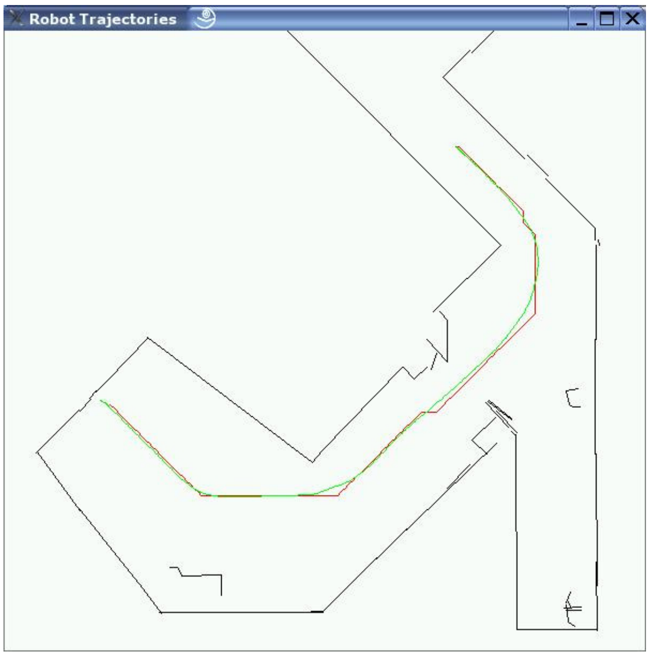


Figure 5.23: Trajectory of the robot using the ARNL application (Red: ARNL; Green: robot)

and it uses the Bezier curves, as explained in section 4.4.1, to navigate from one point to the following one, until no more points remain on the list.

However, following this approach, the path calculated by the ARNL navigation task has some “sharp” angles which cause the conductor to turn in an abrupt way, which prevents the robot followers from being able to follow it and lose their position in the formation. In figure 5.23, some of these situations can be seen in the path from the ARNL navigation task, in red. To avoid this problem, some modifications are applied in the list of positions to smooth the trajectory that the conductor must follow, and consequently the followers.

The first modification is aimed at smoothing all the turnings present in the path. To this end, all the places where there is a change in the orientation of the position are detected. The position where the turn takes place and all the positions in an interval of 0.5 meters before and after these turns are eliminated. In this way, a smoother trajectory is achieved.

The second modification implies the elimination of superfluous positions in the list, that is, if the robot must follow a straight line, it is not necessary to have several positions on this line. For this reason, the length of the section between each turning and an interval which indicates the separation that must exists between the positions in that section are calculated. The positions that are not at the specified interval are deleted.

In figure 5.24, the previous path calculated by the ARNL navigation task, in red, can be compared with the path after the modifications mentioned. As can be observed, the turns are now smoother.

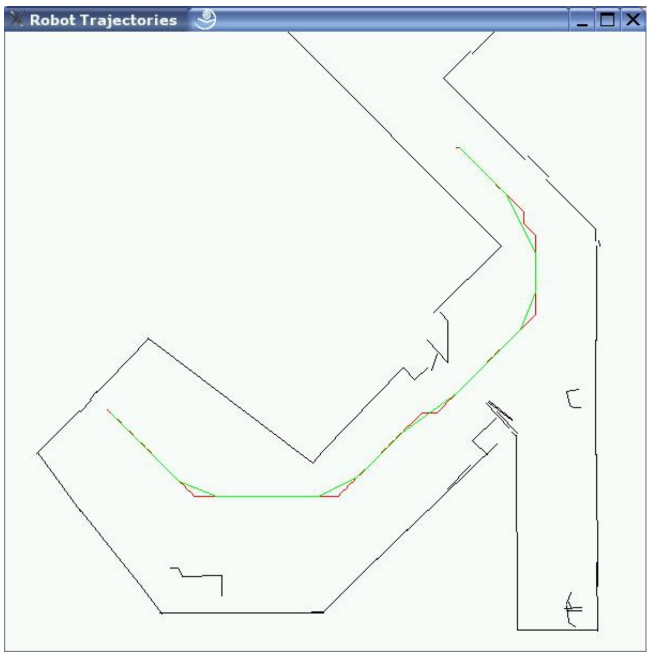


Figure 5.24: Trajectory after the softed process (Red: ARNL; Green: soft)

Finally, in figure 5.25, two examples of the navigation of he robot are shown. In these examples, it can be seen that the robot, after the modification, follows more efficiently the path determined by the ARNL navigation task.

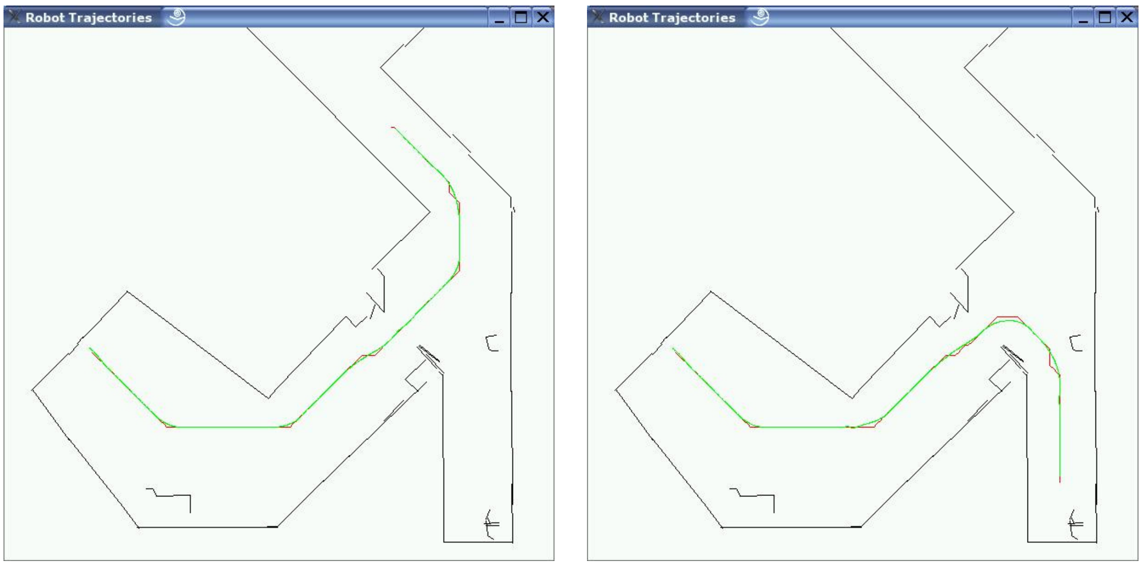


Figure 5.25: Trajectories of the robot using the modified trajectory (Red: ARNL; Green: robot)

5.4.2 Odometry-based formation control

In the first approach to the maintenance of the formation, it was desired that robots could perform the task using only the odometry in conjunction with communication to estimate the position of the conductor with respect to each follower. As explained before, formations of arbitrary shapes can be obtained by the introduction of "virtual points". These points do not correspond to the exact position of the conductor robot, but with an added displacement, as it can be seen in figure 5.21. Thus, the reference for the follower robots is not the conductor robot, but a point located at a given distance at its left or right respectively. Since the position and orientation of the conductor are known, and the displacement of the virtual point is fixed, their estimate is obvious.

The conductor, by means of the LocalizationAgent is always correctly localized in the map, and it is assumed that it will follow a predefined trajectory, with a constant, known linear velocity. In order for another robot to follow the conductor, the linear and angular velocities need to be computed at each time step. It must be noted that the linear velocity of the follower robots is not constant, due to the different radius of their respective trajectories or because their position may be relatively further back or further forward in the formation. Thus, the linear velocity can be computed by simply adding a gain factor proportional to the distance to the conductor robot. The angular velocity can be computed by means of the curvature of the Bezier trajectory in the origin position (P_0). These velocities are updated at each time step, when the new position of the conductor is sent to all the followers.

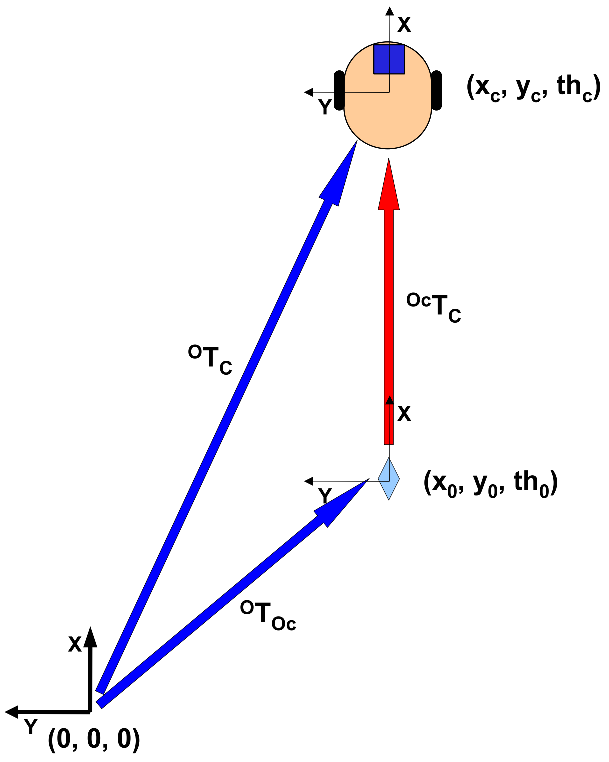


Figure 5.26: Calculation of the local position of the conductor

In order to calculate the linear and angular velocities that allow the followers to move following the movement of the conductor and maintaining the formation, it is necessary to construct the corresponding Bezier curve that defines the trajectory to be followed by the robot. To compute the Bezier curve two positions are needed, the current position of the follower and the current position of the leader in relation to the follower.

The conductor, at each step sends its own position to the followers. As it has been said before, the conductor is always localized in the map by means of the LocalizationAgent, so, when it requires information about its position, this is done with the coordinates of the map. The followers are not localized on the map, so the only information they have available is their position on their own local system which is determined by the origin that is fixed by their initial position. So, the conductor must transform their global position into a position in its local system. In figure 5.26, the necessary relationships to transform the global position into a local position can be seen.

From the figure, it can be deducted that to calculate the position in the local system it is necessary to know the origin of the trajectory in the global system and the actual position of the robot in the global system.

$$^{Oc}T_C = \left(^{O}T_{Oc}\right)^{-1} \cdot^{O} T_C \tag{5.15}$$

where $^{Oc}T_C$ is the actual position of the conductor in the local system, $^{O}T_{Oc}$ is the origin of the trajectory in the global system, and $^{O}T_C$ is the actual position of the conductor in the global system. This last position is updated continuously by the Localization task so that the robot have always its current position on the map.

The conductor sends its current position calculated in this way to each follower. When the followers receive the position of the conductor, they need to calculate the position of the conductor in relation to themselves in order to use that information to generate the Bezier curve necessary to calculate the velocities which allow them to move maintaining the formation.

Each follower knows its original displacement in the formation. This information in conjunction with the current position of the follower and the current position of the conductor allows the robots to calculate the relative position of the conductor in relation with the follower. In figure 5.27, the necessary relationships to compute the conductor position are shown.

In this case, from the figure, the following equation to calculate the conductor position in relation with the follower $^{F}T_C$ can be obtained,

$$^{F}T_C = \left(^{Of}T_F\right)^{-1} \cdot^{Of} T_{Oc} \cdot^{Oc} T_C \tag{5.16}$$

where $^{Of}T_F$ is the position of the follower in its local system, $^{Of}T_{Oc}$ is the position of the origin of the conductor's trajectory in relation to the origin of the follower's trajectory. This value is indicated by the displacements assigned to the position of each robot in the creation of the formation. And finally, $^{Oc}T_C$ is the position of the conductor in its local system. This is the position that the conductor sends every step to the followers.

Once this position is calculated, it is possible to compute the Bezier curve between the current position of the follower and the position calculated for the conductor. It is at this moment when the virtual points are calculated, if applicable, applying the following formulas,

$$\begin{aligned} x &= x - \sin(\delta) \cdot dy \\ y &= y + \cos(\delta) \cdot dy \end{aligned} \tag{5.17}$$

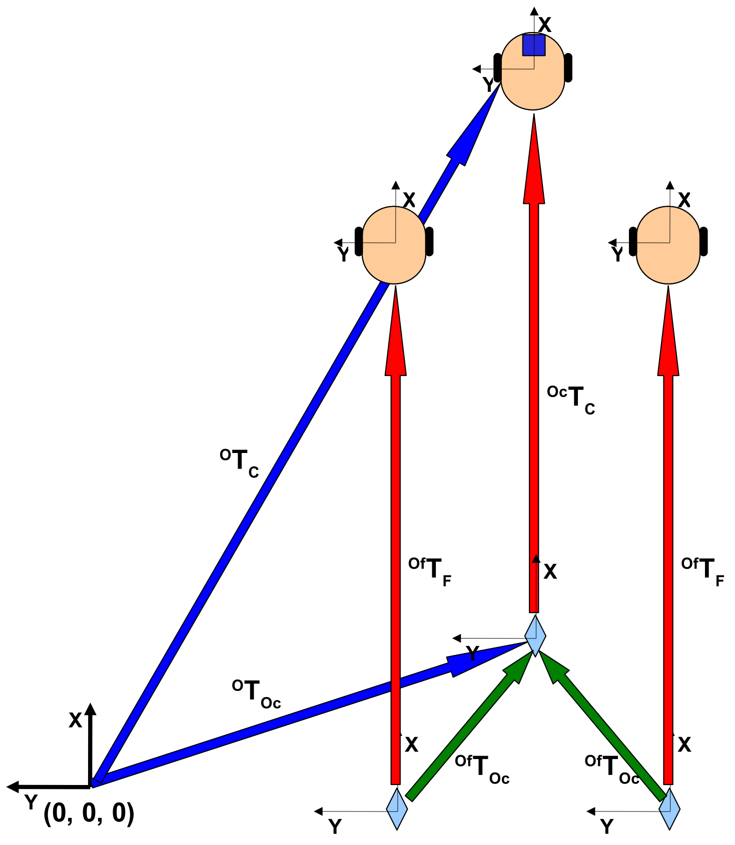


Figure 5.27: Calculation of the conductor's position in relation to the followers

where (x, y, δ) is the position of the conductor in relation to the follower, and dy is the displacement in the y axis to be applied. This generates a new point, the virtual point, which will be used to compute the Bezier curve that drives the movement of the follower.

From the Bezier curve computed, the linear and angular velocities can be obtained. The angular velocity, as it was said before, can be computed from the curvature of the curve, and the linear velocity is computed in order to maintain the distance from the conductor, applying a gain factor proportional to the current distance from the conductor robot. If the robots is nearer than a predefined distance in the formation, it will move slower, and if the robot is further than the predefined distance, it will move faster.

This approach has been tested in simulation and in real robots with different results in both cases. In figures 5.28, 5.29, and 5.30, some examples executed using the simulator for different formations shapes are presented together with the formation in which the robots are organized. Formations from two robots up to four robots can be used.

As it can be seen, the odometry-based approach explained in this section is suitable for the development of applications using multirobot formations. From the different examples, it can be infered that the behaviour of the robots is as desired, and all the follower robots can follow the movements of the conductor robot by the utilization of virtual points. Moreover, it can be noticed that the robots follow the predefined path imposed by the conductor while maintaining the formation at all times.

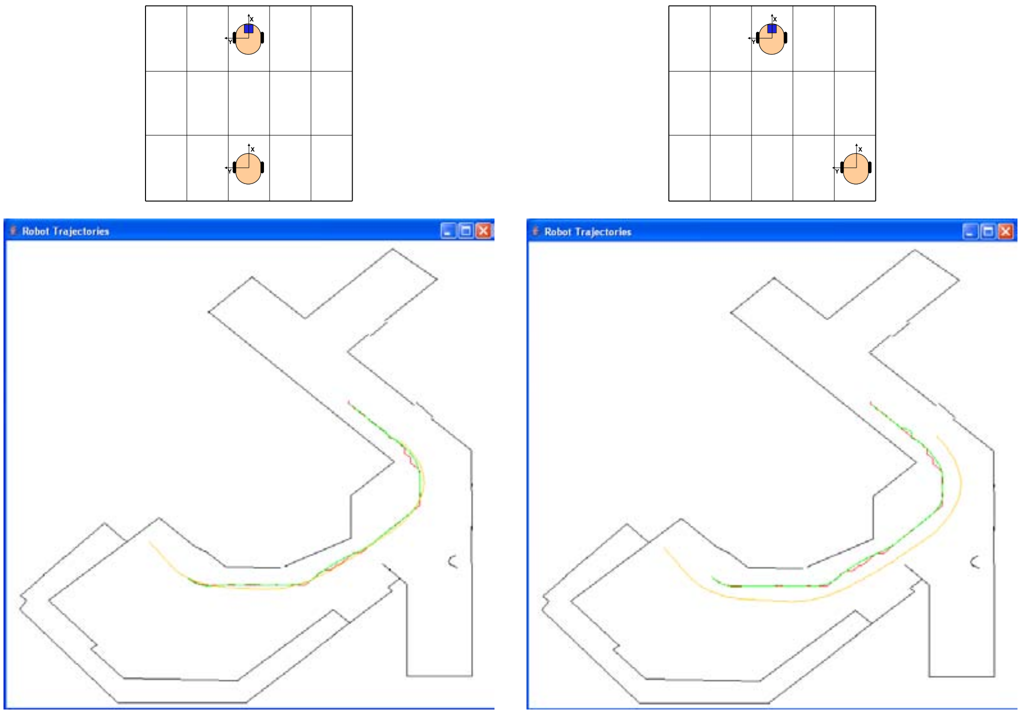


Figure 5.28: Examples of formations with only two robots, the conductor and one follower

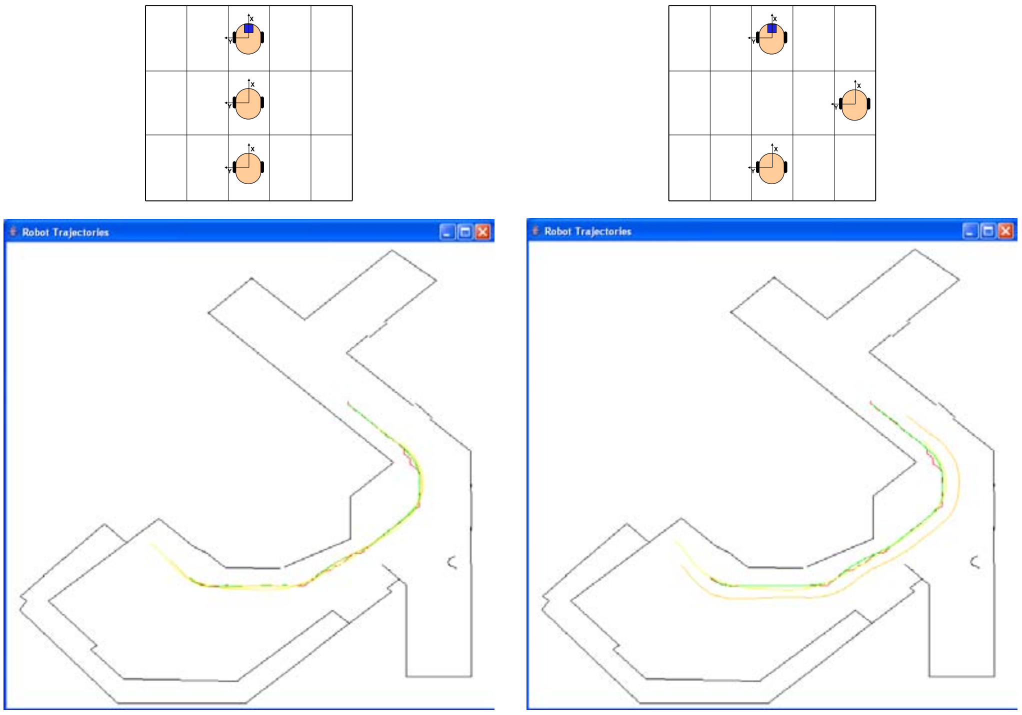


Figure 5.29: Examples of formations with three robots, the conductor and two followers

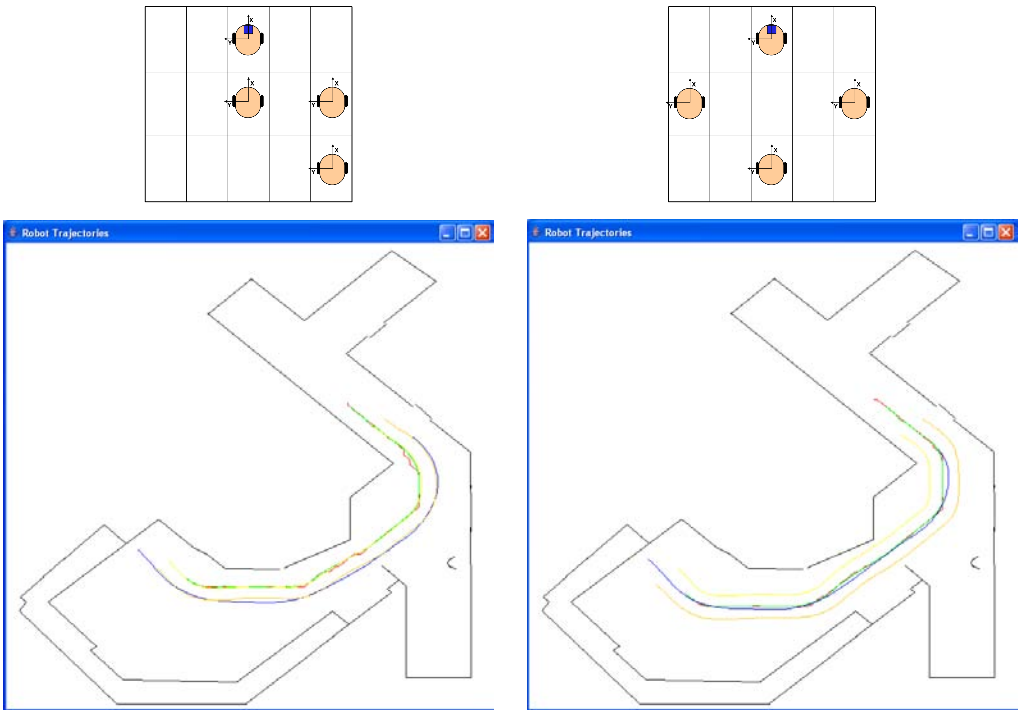


Figure 5.30: Examples of formations with four robots, the conductor and three follower

After testing in the simulator with different formation shapes and verifying that the approach produces the desired behaviour, it was the turn of testing that approach in real robots. The change from the simulation to the real robots is automatic, because no modification in the system or in the code is needed in order to execute a program that has already been used in the simulator.

However, after several tests, it was noticed that something was wrong in the execution in real robots. In the simplest case of formation, where one robot must follow the conductor, the follower was not able to follow the conductor as desired. As it can be seen in the photographs of figures 5.31 and 5.32.

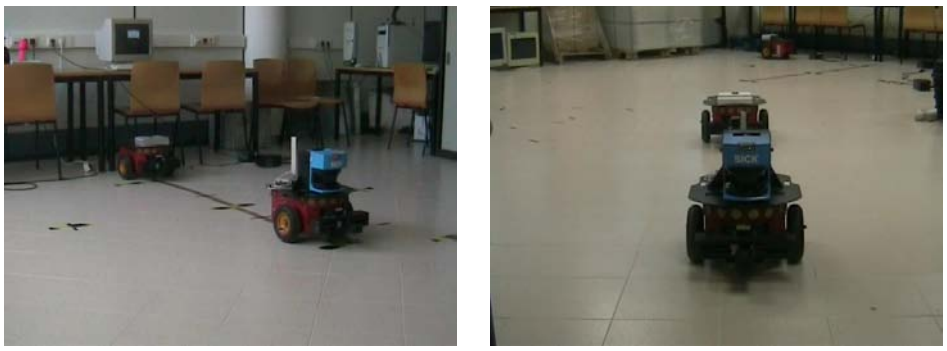


Figure 5.31: Execution in real robots with odometry errors in the navigation (1/2)

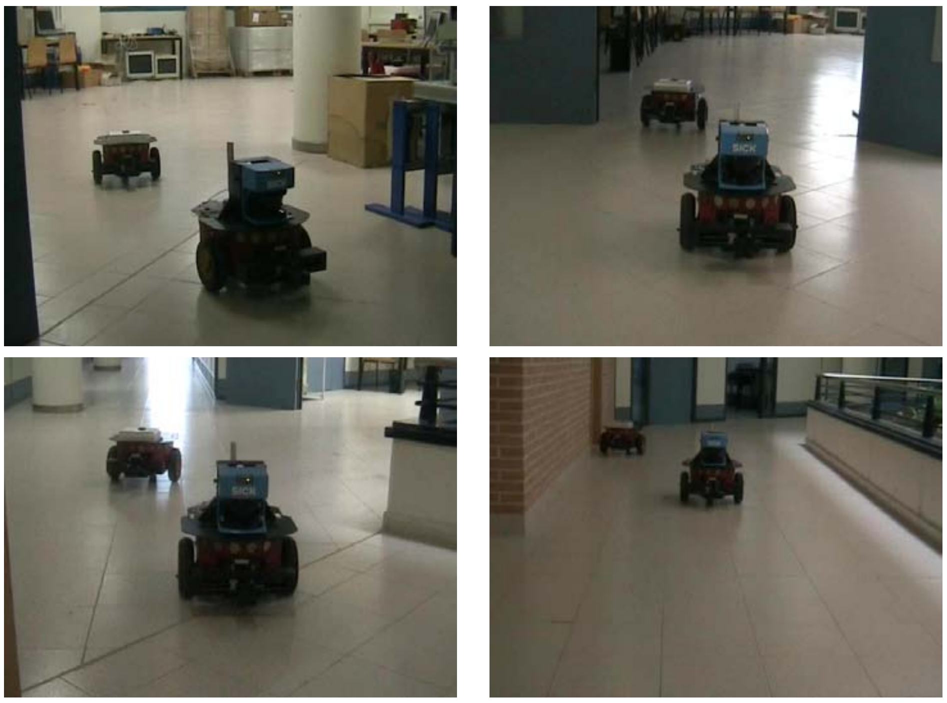


Figure 5.32: Execution in real robots with odometry errors in the navigation (2/2)

It was noticed that the problem was due to the odometry errors of the robots. In the simulation, no odometry errors were presented, but when the application was executed in real robots, these errors meant that it was not suitable for the execution in real robots. These odometry errors influence in the behaviour of the follower, making it impossible for it to follow the conductor maintaining the formation.

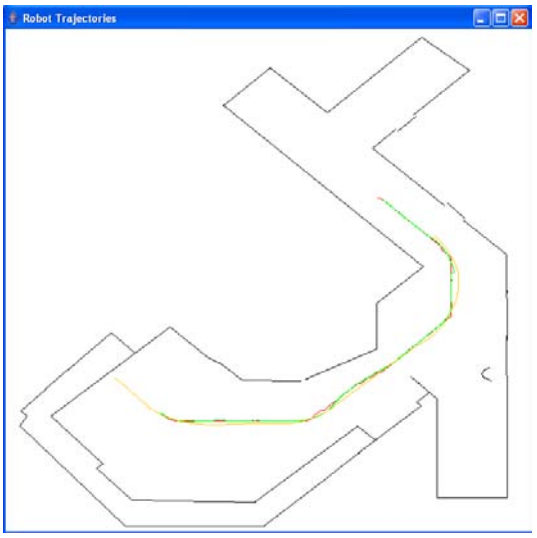
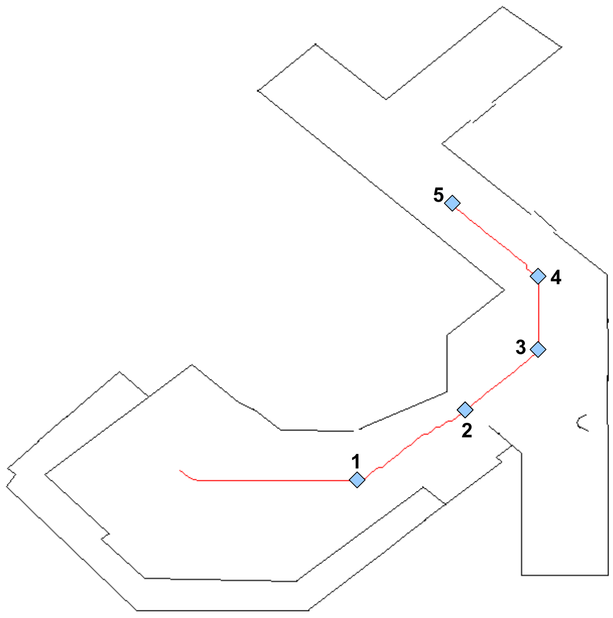


Figure 5.33: Robot believes that its position is the correct one

However, the problem of the odometry errors does not influence the way that the robot perceives its own position. That is, the robot thinks that it is in the correct position, but really, the robot is not in the position that it thinks, making it impossible for the application to take any steps to solve these errors while the robot is moving. This behaviour can be checked comparing the figure 5.33 with the photographs in figures 5.31 and 5.32.

The errors of odometry grow with the distance covered and can reach unacceptable levels. Some measurements have been made in order to know the importance of this error in the navigation. These measurements have been taken manually at some predefined points in the path followed by the robots. The points where the measurements were taken and the measurements are shown in figure 5.34, and the measurements taken are shown in the table.



Position	Measurement
1	(3cm, 5cm)
2	(-10cm, 12cm)
3	(-25cm, 30cm)
4	(-25cm, 37cm)
5	(25cm, 55cm)

Figure 5.34: Positions where the odometry errors were measured

Implementation of the application

To integrate in the robots the capabilities here described, new agents have been developed. In this special case, only two agents are necessary to incorporate into the robots the navigation by means of the odometry, the ConductorAgent in the conductor robot and FollowerAgent in each one of the follower robots.

ConductorAgent is the robot in charge of leading the formation on a predefined path. In order to calculate the path, this agent communicates with NavigationAgent and by means of ARNL it is possible to calculate the path to follow in a map from one origin point to a goal. While this agent drives the conductor by the path calculated, it must send its actual position to each follower at every step, as it was explained before. Also, in order to get precise information about its own position in the map continuously, this agent uses the LocalizationAgent for this purpose.

On the other side, *FollowerAgent*, when it receives the information about the actual position of the conductor, calculates the virtual point, if needed, and the Bezier trajectory to follow the conductor robot maintaining the formation.

Figure 5.35 illustrates the coordination among the different agents in a diamond-shaped formation using only the odometry for the maintenance of the formation.

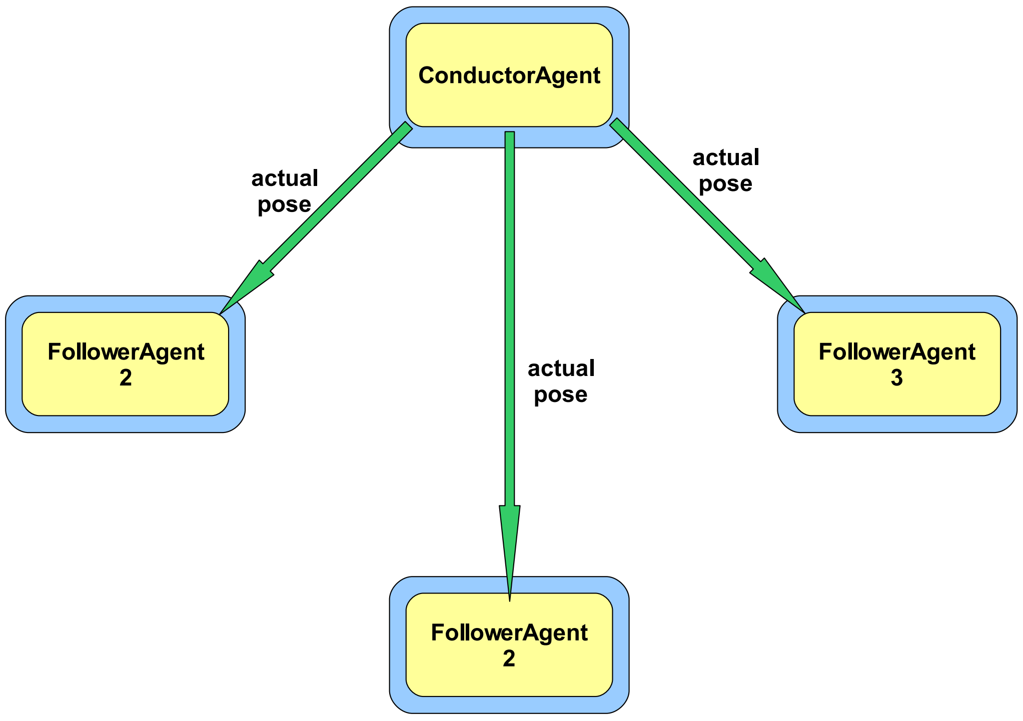


Figure 5.35: Coordination among the agents in the odometry-based formation control

5.4.3 Vision-based formation control

To solve this problem, the decision was to use the camera of the leader robot to improve the navigation of the robots and reduce the odometry errors. Monitoring the position of the followers while the formation is moving, it is possible to indicate to them when they have lost the position in the formation and what is their current position. The followers, in that case, must perform a modification of their trajectory in order to put themselves in the correct position in the formation once again.

In order to calculate those real positions, the leader uses the targets that each follower carries and the target that the conductor also carries. In this approach, the leader must be at the back of the formation and all of the followers and the conductor being visible, and uses the camera as explained in section 5.3 to localize each one of the robots.

Determination of the position and orientation of any robot can be achieved by estimating its distance and relative orientation with regard to the leader robot. The position of the observed robot can be obtained from its estimated distance and the pan angle of the camera. The relative orientation of the robot can be obtained by means of the observation of the target.

Because the leader is also a follower, its position must be calculated, and for this, it uses the target of the conductor. Localizing the position of the conductor it is possible to establish the real position of the leader. So, two possibilities in the calculation of the position of the followers are implemented.

In the case of the calculation of the position of the leader, it simply localizes the position of the conductor, and by means of this position, its real position can be calculated. Thus, to calculate its position, the leader needs to know three values, the position of the conductor in relation to itself, ${}^{L}T_{C}$, the current position of the conductor in its local system, ${}^{Oc}T_{C}$, and the position of itself in relation with the conductor in the creation of the formation, ${}^{Oc}T_{Ol}$. Knowing that the current position of the conductor in its local system can be calculated as

$$ {}^{Oc}T_{C} = ({}^{O}T_{Oc})^{-1} \cdot {}^{O} T_{C} \tag{5.18} $$

the actual position of the leader is calculated as

$$ {}^{Ol}T_{L} = ({}^{Oc}T_{Ol})^{-1} \cdot {}^{Oc} T_{C} \cdot ({}^{L}T_{C})^{-1} \tag{5.19} $$

In figure 5.36, this calculation can be seen graphically.

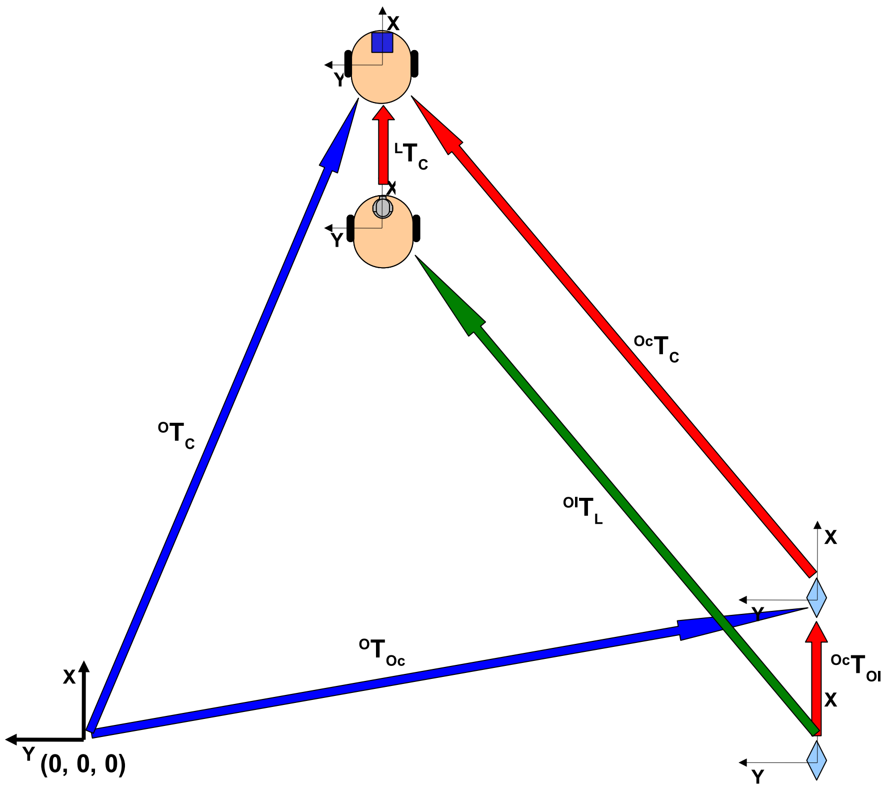


Figure 5.36: Calculation of the conductor's position in relation to the followers

In order to calculate the position of any of the rest of followers, it is also necessary to know the position of this robot in relation to the leader, which is calculated by the leader using the camera. Thus, to calculate its position, the follower needs to know four values, the position of the follower in relation to the leader, ${}^{L}T_{F}$, the position of the conductor in relation to the leader, also calculated with the cam, ${}^{L}T_{C}$, the current position of the conductor in its local system, ${}^{Oc}T_{C}$, and the position of the follower in relation to the conductor in the creation of the formation, ${}^{Oc}T_{Of}$. Knowing that the current position of the conductor in its local system can be calculated using equation 5.18, the actual position of the follower is calculates as

$$^{Of}T_F = (^{Oc}T_{Of})^{-1} \cdot ^{Oc}T_C \cdot (^{L}T_C)^{-1} \cdot ^{L}T_F \tag{5.20}$$

In figure 5.37, it can be graphically seen.

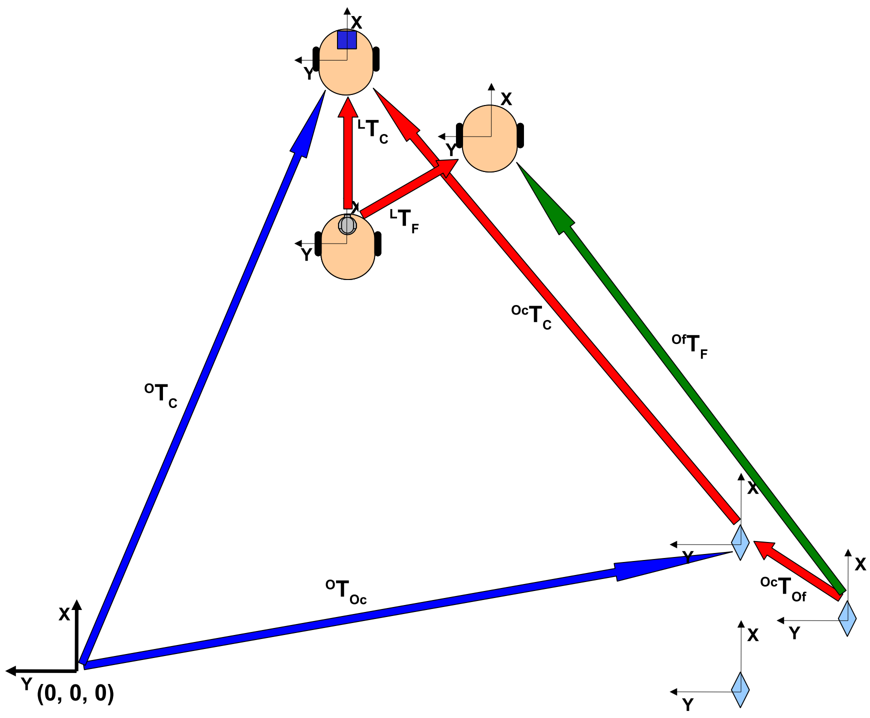


Figure 5.37: Calculation of the conductor's position in relation to the followers

Once one follower has calculated its real position, it changes the value of its actual position, that is, its belief about the position where it thinks it is in that moment. In this way, in the following step, when the follower calculates the Bezier trajectory to follow the conductor, it will use the real position in those calculations, and automatically, following the trajectory of that Bezier curve, it will go to its correct position in the formation while it moves following the conductor.

Some experiments have been performed in order to validate the new approach. Based on these experiments, it can be concluded that the maintenance of the formation has been improved considerably, and now the robots are able to maintains their positions in the formation while following the conductor in any path.

In figures 5.38 and 5.39, two experiments using different numbers of robots are depicted. In the case of figure 5.38, only two robots are used in a classical leader-follower configuration in order to contrast this results with the ones in figures 5.31 and 5.32. Figure 5.39 depicts a new experiment with four robots in a diamond formation. As it can be seen, the robots behave as expected.

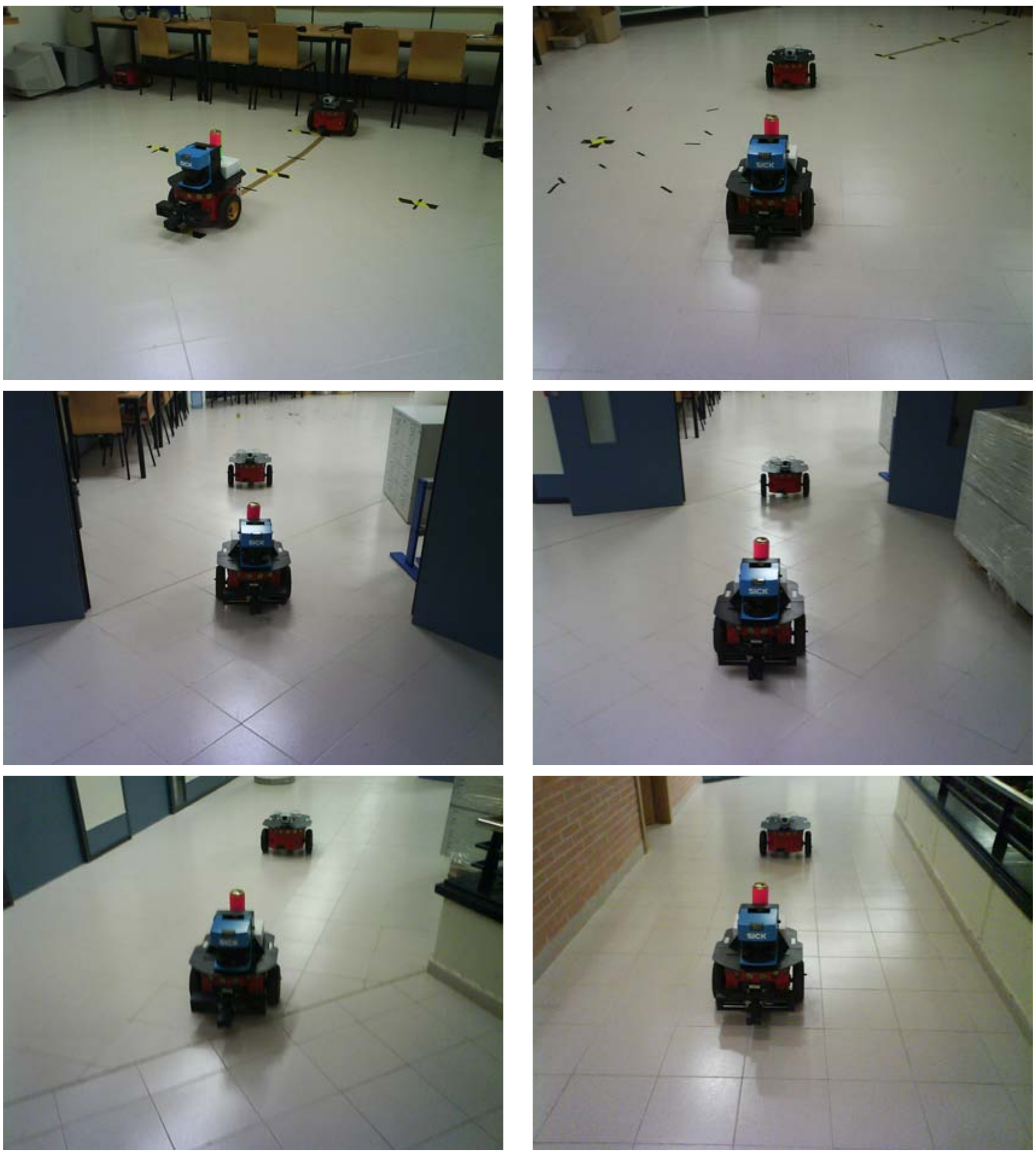


Figure 5.38: Test with two-robot leader-following formation using visual repositioning

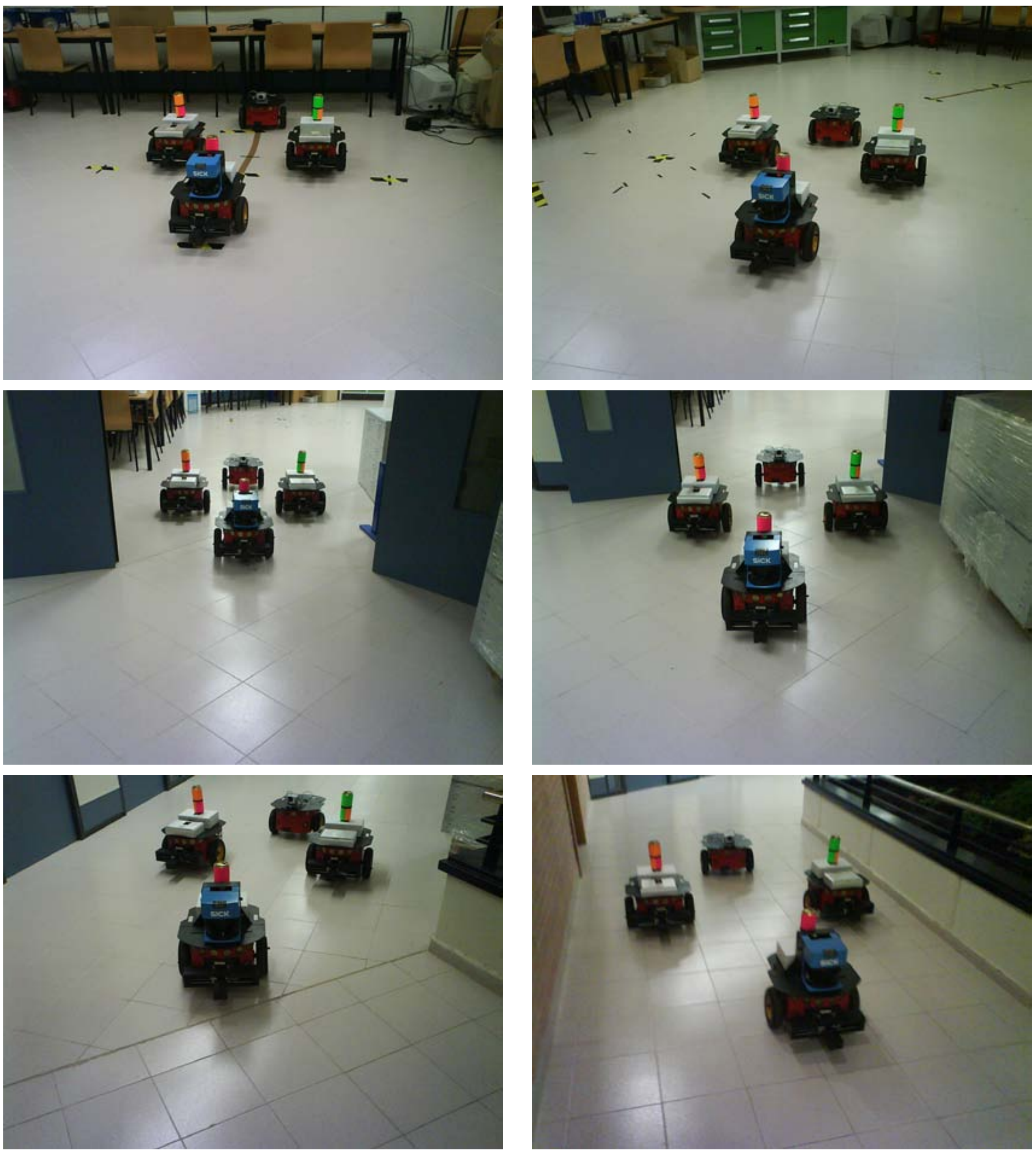


Figure 5.39: Test with four-robot diamond formation using visual repositioning

Implementation of the application

In this case, the introduction of some old agents is necessary. These agents are the ones used before to localize the robots in the environment, TrackerAgent, SearchAgent, CenterAgent and IdentifyAgent. Their function in this case is similar to the function that they carried out before, they must search continuously for each one of the followers, one per one, and calculate their position in relation to the leader robot. Then, this position is sent to each follower to calculate their real position and try to avoid the odometry problems.

At the same time, the ConductorAgent continues sending their actual position to each robot follower in order that they can calculate the necessary trajectory to maintain the formation and follow the conductor robot.

In figure 5.40, the coordination among the different agents in a diamond-shaped formation using only the vision for the maintenance of the formation is shown.

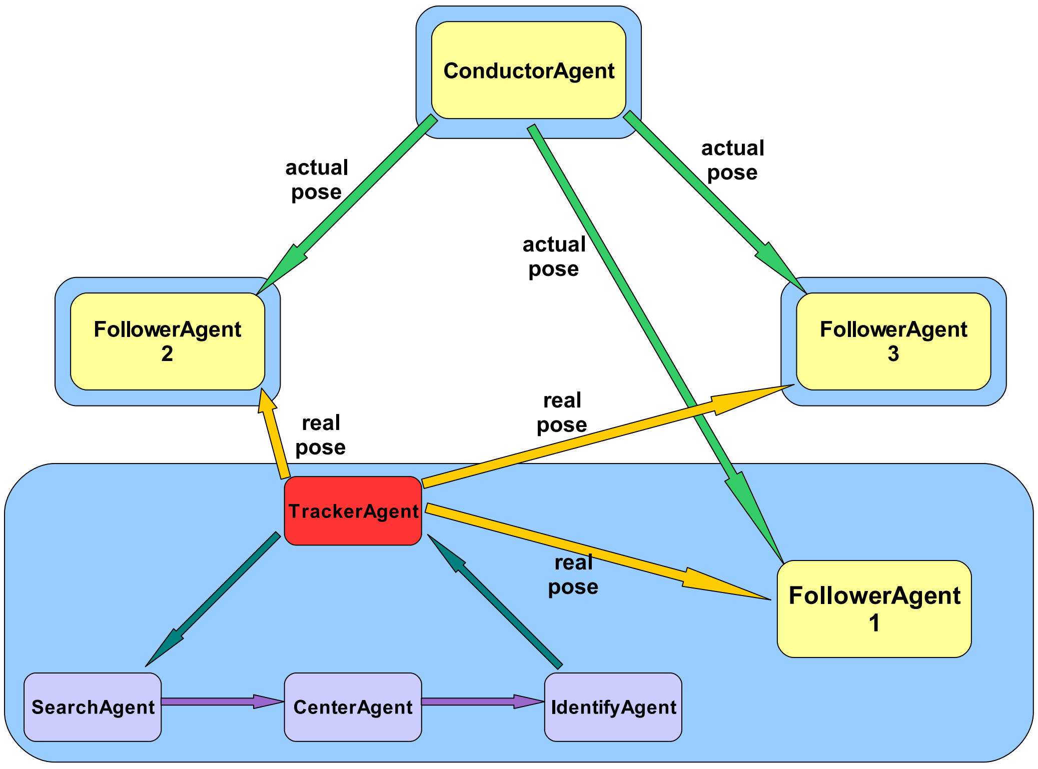


Figure 5.40: Coordination among the agents in the vision-based formation control

Chapter 6

Teleoperation of a team of mobile robots

With the purpose of making the team of robots accesible for any user, independently of their knowlege in robotics, a agent-based teleoperation system is implemented.

6.1 Overview

Completely autonomous performance of a mobile robot with non-controlled and dynamic environments is not possible due to several reasons including environment uncertainty, sensor/software robustness, limited robotic abilities, ... But in assistant applications in which a human is always present, he can make up for the lack of robot autonomy by intervening when needed [139].

Teleoperation is a kind of Human-Robot Interaction (HRI) which is distinguished by having a physical distance between the robot and the human operator which prevents the operator from seeing the robot during the execution of its tasks. It is usually used in cases where the environment is not directly accessible by a human, or is too dangerous, or it is necessary to scale the force exerted on the environment. Therefore it is necessary to rely on information and communication technology for efficient transmission of the operator's commands and of the robot sensory feedback [62].

In a teleoperation environment, it would be to have the possibility to interconnect various devices, test different control algorithms, process data from various sensors and use various connection media [75]. From this assumption, the idea of using a middleware integration framework arises. The Acromovi architecture fulfills all those requirements, making it a good option for the implementation of a teleoperation system.

The main goal of this section is to make teleoperation of a team of mobile robots accessible to all users, novices and experts. So, a "human-robot integration" mechanism is needed to improve the robot autonomy in daily environments. Thus, an easy-to-use user interface and an effective agent-based human-robot interaction method to enable robust mobile robot teleoperation have been created. Moreover, it is desired that the system will be able to cope with the different features of each robot in the team in a transparent way for the operator.

6.2 Previous works

During the years spent for the PhD degree, the author of this thesis has been involved in different projects that had in common the teleoperation of different devices in order to help students and novel researchers in their work. Students used these teleoperated systems as a new tool for learning more about the devices and how they operate, and researchers as a help in their tests of new applications or developments.

The first introduction of the author to the world of teleoperated systems was thanks to Raul Marin and Pedro Sanz who counted on me to help them implement some new features for the UJI Robotics Telelab. These first tasks were devoted to implementing a new concept of teleoperation that is remote programming.

Until then, the teleoperation of the UJI Online Robot had been done by means of various methods such as direct robot positioning, high level commands or voice commands. With the introduction of the remote programming in the system, a new point of view in the commanding of the robots is stablished. Both students and researchers have benefitted enormously from this project because now they can program the system remotely to perform an autonomous task and validate their algorithms in the remote robot without the necessity of being present in the place where the robot is set up. In figure 6.1, it is possible to see the result of this research as the introduction of a new level in the architecture that allows the operator to program the system remotely.

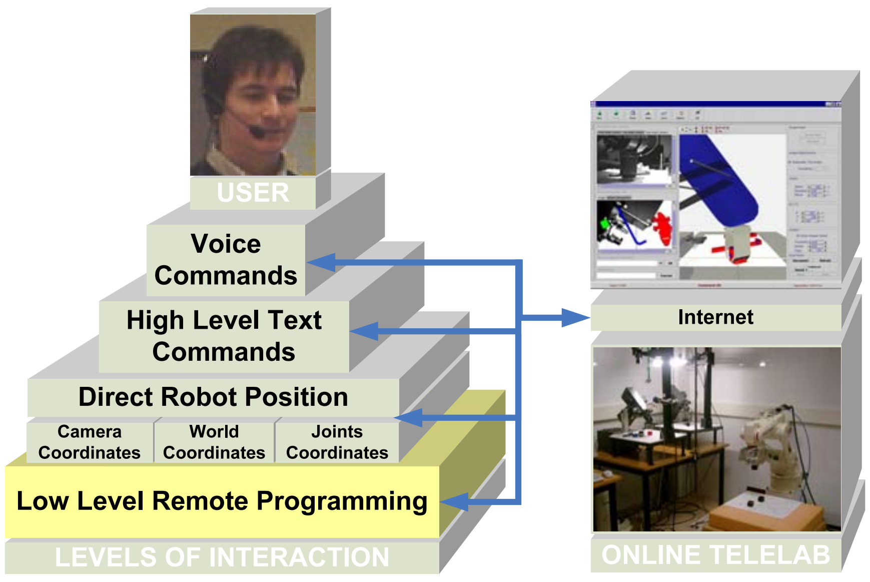


Figure 6.1: Remote programming architecture over the UJI Online Robot

As result of this research, many papers were published in different conferences and journals [226, 227, 228, 229]. It also serves as one of the research projects for obtaining the Advanced Studies Diploma while studying the PhD degree.

All these previous works were focused in the teleoperation of manipulators. But after these experiences, new teleoperated systems for controlling the team of mobile robots were implemented. As an attempt to improve the way in which students learn robotics and trying to reduce the time taken by the researchers to learn how a new system works, a very simple teleoperated system was developed to control the Pioneer-2 robots. It consisted of a graphical interface containing a menu with the different orders that the user was able to execute with the robot. By means of this system, the learning time was reduced considerably. The interface can be seen in figure 6.2.

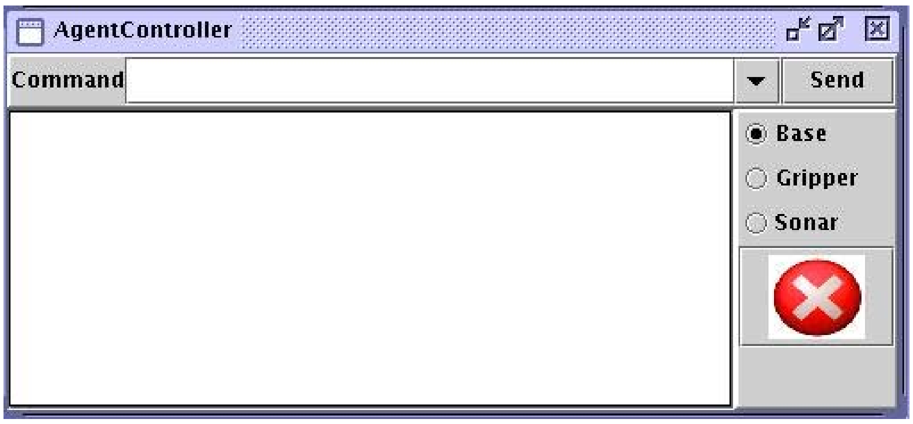
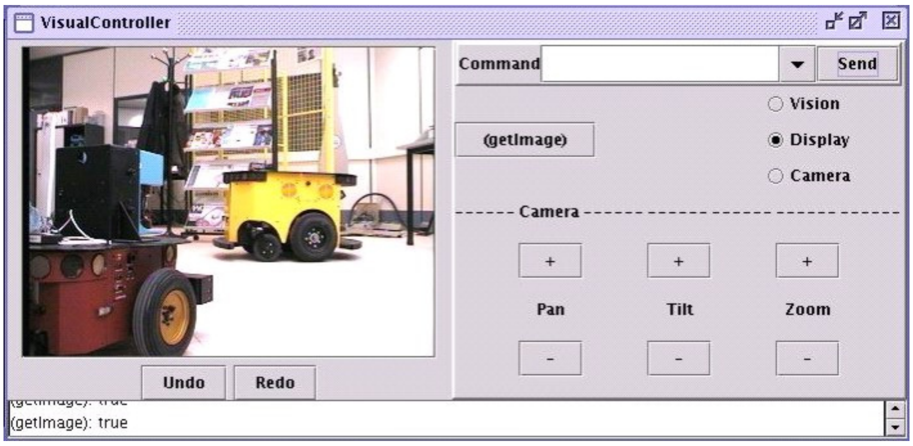


Figure 6.2: Simple teleoperated system to control a mobile robot

From this system, a student, Raul Perez, developed a system to execute vision orders in a robot with a camera. In this way, the programmer could check their actions before implementing the application, thus reducing the time necessary for debugging the programs. The application consisted of a graphical interface, as it can be seen in figure 6.3, which allows the user to send commands to the vision and camera agents and displays the results.



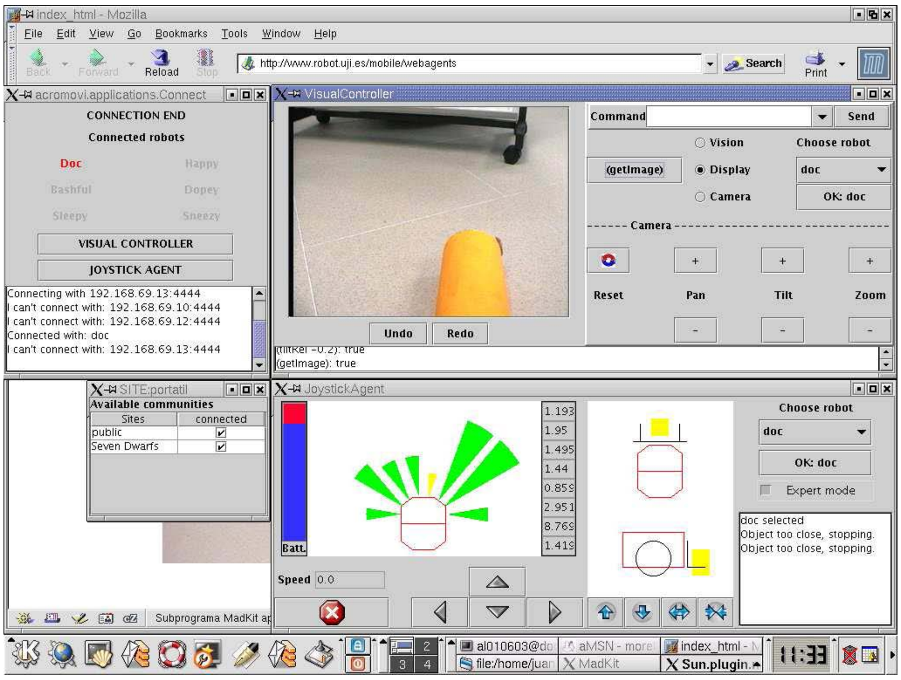
(getImage): true

Figure 6.3: Teleoperation system to perform visual operations with the robots

Such commands include the capture and display of the current image grabbed by the robot camera, the pan/tilt/zoom motions of the camera, and a set of image processing operations. Operations provided were lowlevel operations like thresholding, blurring, edge detection, as well as high-level ones as colour or motion tracking.

An extension of this work was developed by another student, Juan A. Querol, who integrated this visual remote controller and other facilities in a web interface in order to make the teleoperation accessible over the Internet. This interface was implemented as a Java applet in order to have an application which can be launched directly from a web browser without installing any other software in the local machine.

The interface is divided in three modules, one for the connexion with the robots, another one for commanding the robots, called JoystickController, and finally, another module implementing the visual controller seen before. These three modules can be seen in figure 6.4.



```
Connecting with 192.168.69.13:4444
I can't connect with: 192.168.69.10:4444
I can't connect with: 192.168.69.12:4444
Connected with: doc
I can't connect with: 192.168.69.13:4444
```

```
(tiltRel -0.2): true
(getImage): true
```

```
doc selected
Object too close, stopping.
Object too close, stopping.
```

Figure 6.4: Web-based teleoperation system for mobile robots team

Finally, in order to test these systems with real students, they were used for two years in the International UJI Robotics School for carrying out practice activities [270, 267]. These systems served as a first approximation to the control of the real devices that students used to program a specific application. With these applications students learned the set of instructions that hey had to use in order to program the robots and the effects that those orders had on the robots. This fact meant that students did not lose much time in the learning of the system and applied almost all the time in the development of the specific application. Moreover, with these systems, they were able to test some of the parts of their programs in order to debug them in a quick and efficient way.

6.3 Development of the teleoperation system

This section describes the implementation of the control of the teleoperated system, in this case a supervisory control model. This model of control enables the operator to view the current sensory information of the system and allows any robot to be controlled remotely by sending it specific orders or simply by supervising the work that the robot is performing. It is important to remark that control and communications among the human operator and the robots are embedded within the agent-based Acromovi architecture. So, new agents are created which are then integrated into the new application in the system.

This section also describes the human-robot interface implemented in order to operate the robots. The design and operation of graphical user interfaces is key to supervisory control of mobile robots. At this point, a multisensor/multimodal model has been used to implement the interface. The interface has been implemented following the guidelines of JADE, resulting in an easy and fast integration of a graphical interface with the agent system. Also, Java graphical libraries, like Java AWT (Abstract Window Toolkit) and Java Swing, have been used to make the interface more user friendly.

6.3.1 Control of the teleoperation system

The control model implemented uses the Acromovi agent-based architecture to connect the different parts of the system. It is implemented using agents and the JADE software which facilitates the message sending among the different parts of the system. The user, who is connected to the system via the user interface, is able to send orders to any robot and to see the sensory information of the desired robots in the interface to check the consequences of those orders.

In the remote site, that is, in each robot, there is a group of agents which represents their elements and allows any other agent or application to access to their facilities, to get information or perform actions.

In the local site, there are two different agents that communicate with the agents of the remote site, that is, with the agents of the robots. The first agent is the agent that controls the whole application and facilitates the communication with the different robots of the system. It is called MainAgent. The other agent is in charge of communicating directly with the agent of the robot assigned to it. This agent is dynamic so that it takes different forms depending on the robot to which it is assigned. This agent is called RobotAgent. It is also in charge of getting all the sensory data from the robot and sending the orders generated in the graphical interface.

MainAgent

This agent is in charge of controlling which robots are connected and sends this information to the main window of the interface. To know which robot is connected at any given moment, this agent sends a message to the BaseAgent of any robot asking for its battery level. If it gets the answer of any robot, it indicates that the robot is connected, so that information can be transmitted to the interface, allowing the user to communicate with that robot. This agent is periodically carrying out this process and updating the interface with the results. It is also possible to perform this process whenever the user needs it by a simple interaction with the interface which is detected by this agent.

Moreover, this agent is responsible for instantiating a new RobotAgent for each connected robot. If the user wants to communicate with a specific robot, it must indicate this by means of the graphical interface, and at that moment, if the agent was not created previously, the MainAgent creates a new agent in charge of communicating with the robot in question. This new agent, as it will be seen next, will communicate with the agents of the robot to get all the necessary sensory information and to send the orders that the user want to execute with that robot.

RobotAgent

In order to communicate with the agents of a specific robot, a new agent called RobotAgent has been implemented. The function of this agent is to request information to update the interface and to send orders that the user performs through the graphical interface.

This agent has been created to be dynamic, that is, depending on the initial configuration of the robot, the agent will take various types of actions. This makes the management of the heterogeneous nature of the robot team more user friendly and transparent for the user or operator.

Depending on which robot the agent represents and the elements that the robot has available, it will take different actions in order to update the graphical interface and a different set of orders will be available to interact with the robot.

This agent has been implemented by means of a compound parallel behaviour compose of three or four behaviours, depending on the configuration of the robot, which will be run concurrently. Each behaviour will communicate cyclically with a different agent of the robot, which represents an element of the robot, requesting information on the state of the corresponding element to update the interface.

This agent is also in charge of sending the agents of the robot the action that the operator indicates in the interface. By means of events generated in the interface, the agent can detect the action performed by the user and send the proper action to the specific agent in the robot.

As result of these actions, the user can see the results of their actions, soon after, thanks to the information gathered by the cyclical behaviours in the parallel behaviour.

In the figure 6.5, an overview of the control system and the relation among the agents is shown.

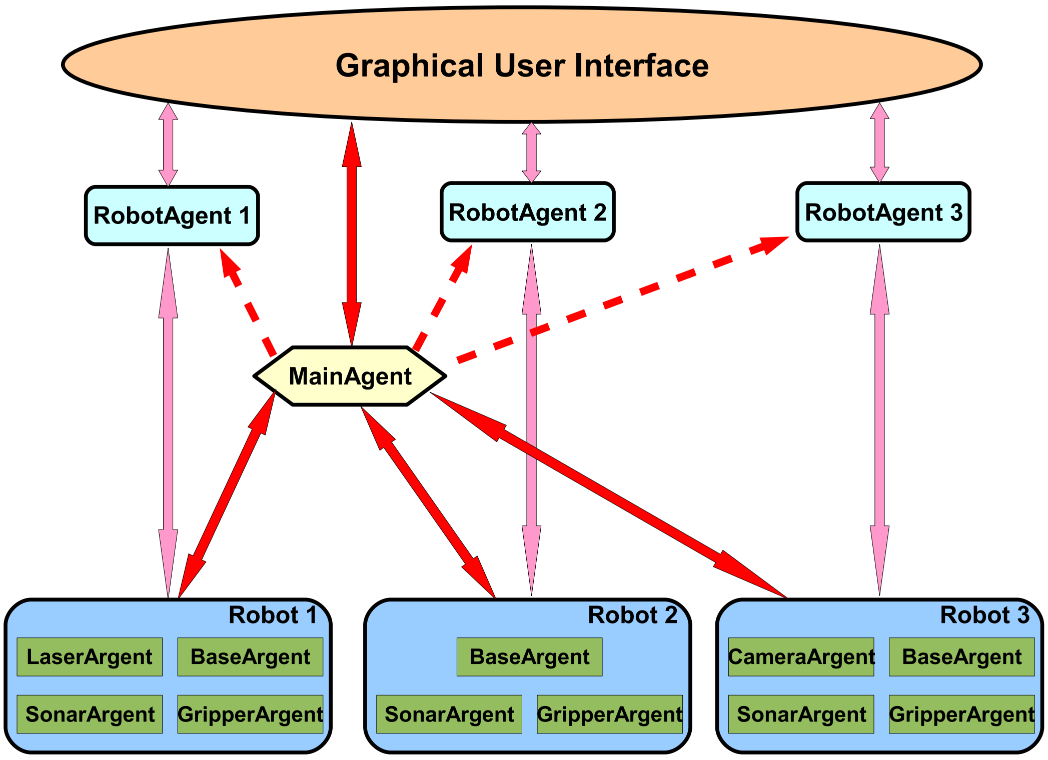


Figure 6.5: Control model for the teleoperation

6.3.2 Design of the graphical user interface

To improve the teleoperation, it is necessary to make it easier for the operator to understand the remote environment and to make control decisions. That is, it is necessary to maximize the information transferred while the cognitive loading in the design of the human-robot interface is minimized. The system provides the operator with rich information feedback, facilitating the understanding of the remote environment. In this way, sensor fusion displays combine information from multiple sensors or data sources into a single integrated view [130].

Mobile robot teleoperation interfaces are often difficult to use and require extensive training. As a result, only experts can achieve an acceptable performance. In order to make mobile robot teleoperation accessible to all users, whether they are experts or not, it is necessary to make the interfaces easy to deploy, easy to understand and easy to use. For this reason, the interface must make understanding the remote environment easy and generating motion commands for the robots efficient and accurate.

The interface is designed to take into account these features and consists of a series of displays, one per connected robot, which contain the sensor information and a variety of command generation tools. Moreover, due to the heterogeneity of the robots in the system, as it was seen in section 3.2.1, these displays will vary depending on the robot that they are representing.

The interface consists of two different types of windows, the main window and the dynamic robot windows. The main window enables the user to control the whole interface and the communications with the different robots. It also is used to supervise the work when the robots are performing a cooperative task. In the robot windows, all the information relative to a specific robot is displayed. As the robots are heterogeneous because of the differences of their accessories, these windows will show different information depending on the robot. It is important to remark that these differences are dealt with in execution time in a transparent way to the user.

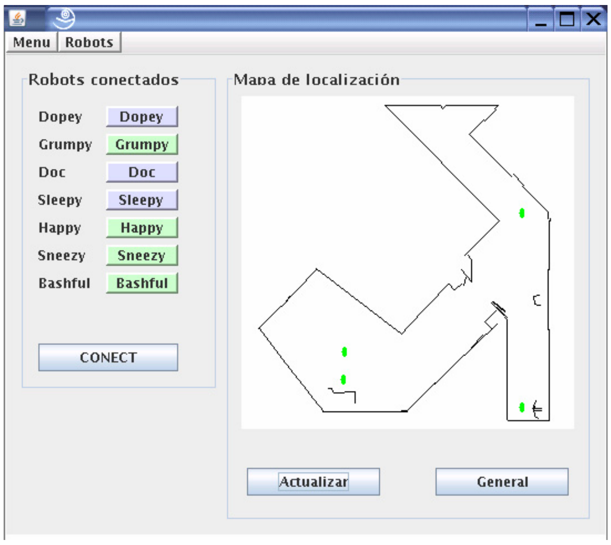


Figure 6.6: Main window of the interface

Main window

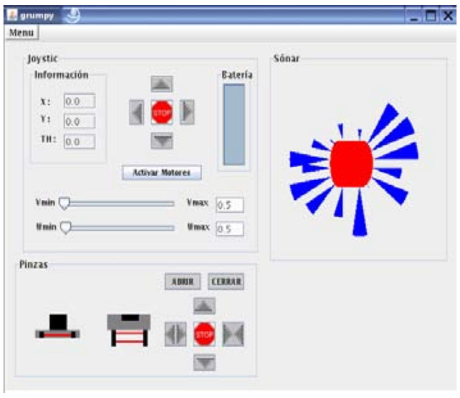
Figure 6.6 shows the main window of the interface. This window consists of three differentiated displays: (a) an applications menu, (b) a panel with a list of the robots and (c) a map for the localization of the robots.

The display with the list of robots is composed of a button for each robot that allows access to the robot window. This buttons have two different states, depending on the state of the robots, that is, if the robots are active or not. If the robot is active, the corresponding robot window will appear when pressing the corresponding button.

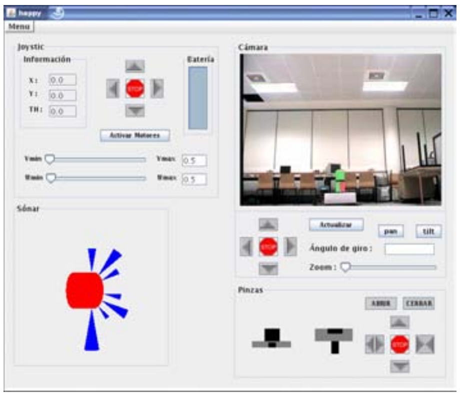
On the map, the robots which are connected and have their position localized in a map are shown in the proper position. Finally, in the menu the same actions that can be executed with the graphical interface are displayed.

Robot windows

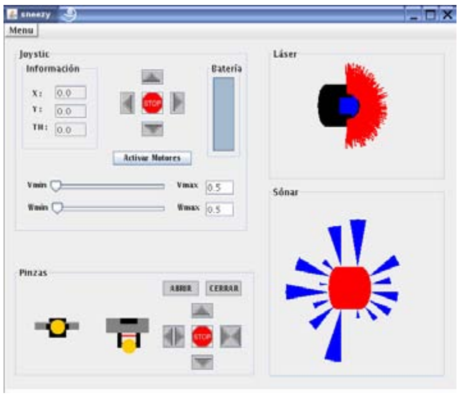
These windows have been designed by composing a set of three or four displays depending on the configuration of the robot. In figure 6.7, all the different possibilities for the appearance of these windows can be seen.



(a) Dump robot window



(b) Camera robot window

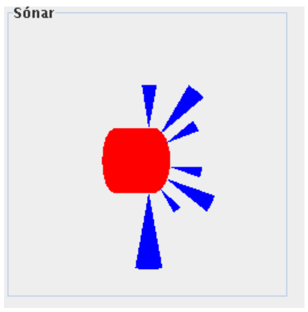


(c) Laser robot window

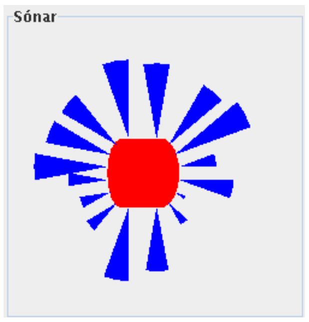
Figure 6.7: Different robot windows depending on the configuration of the robot

As can be seen in the figure, there are three possible configurations of the window corresponding with the three configurations of the robots. If the robot only has the sonars and the gripper, the window will take the (a) configuration. If the robot also has a vision system with a camera, the configuration is the (b) option. Finally, if the robot has a laser system, the configuration is the (c) option. Now, a brief description of every display in the robot window is provided.

The sonar display is only for collecting information about the sonar sensor. It consists of a representation of the robot and over that representation the different values of the sonar in a graphical representation. The sonars are represented by a group of cones forming a semicircle or a whole circle, depending on the robot, which represents the eight or sixteen sonar sensors, as can be seen in the figure 6.8.



(a) Front sonar



(b) Both sonars

Figure 6.8: Sonar display depending on the number of sonars

The sonar cones vary their length depending on the value of the sonar disc sensor associated. In this way, it is easy to see for the user which are the values of the sonar and to take the corresponding action if an obstacle is detected.

The laser display represents the robot with the 180 laser beams of the laser sensor, as it is shown in figure 6.9. As in the case of the sonar display, the length of the laser beams vary depending on the values of the laser device, facilitating in that way the identification of obstacles in the way of the robot.

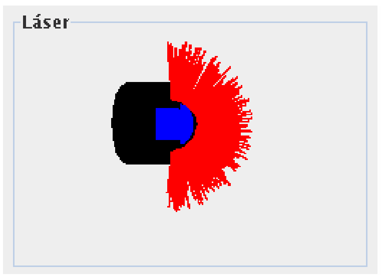


Figure 6.9: Laser display

The joystick display consists of four different sections, as it can be seen in figure 6.10. The first section is formed with the joystick buttons used to handle the direction of the mobile robot. The user controls the robot's forward/backward speed by clicking on the vertical axis. Clicking on the horizontal axis controls the robot's rotation. Also, a button to stop the movement of the robot is provided. The second part contains two swing bars to modify the linear and rotational velocity of the robot, which affect the buttons of the joystick. In the third part, the battery level of the robot can be seen. And finally, the fourth part is a small table where the information about the position of the robot is shown.

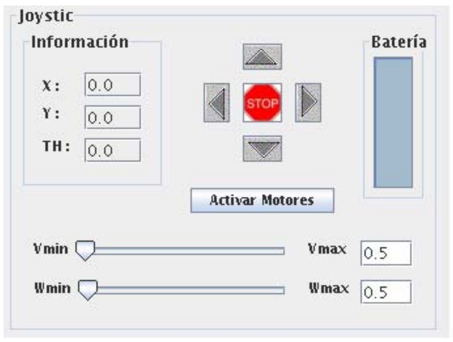


Figure 6.10: Joystick display

The camera display reproduces live images from the camera located on the robot. Also, a set of joystick buttons which the user can use to pan and tilt the camera by clicking on them and a swing bar to enlarge or shrink the image (zoom) are provided. Moreover, there is the possibility to pan or tilt the camera at a certain angle by means of the pan and tilt buttons. This means that the user can move the camera by two different means: one with joystick buttons that rotates one degree in the direction indicated by the button or indicating an angle and pressing on pan button to turn left or right or pressing on the tilt button to raise or lower the camera depending on the angle being positive or negative. The update button is used to upgrade the image of the camera on the interface when necessary. The whole display can be seen in figure 6.11.

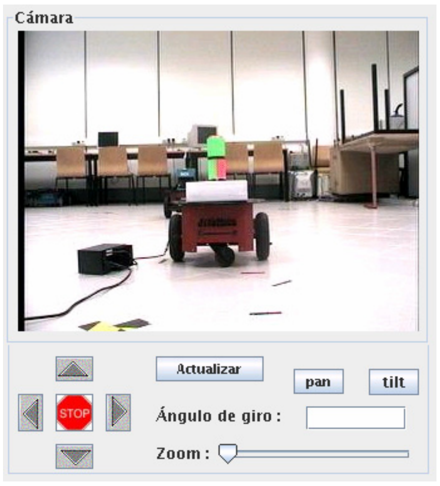
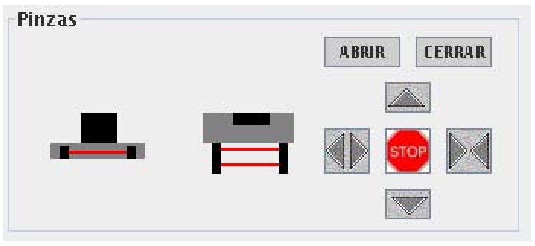
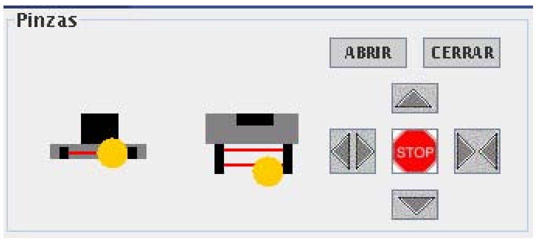


Figure 6.11: Camera display

Finally, the picture of the robot gripper appears in the gripper display in the frontal and upper views to see the status of it (open/closed, up/down, no object/object, ...). It also has a small joystick to open, close, move the gripper up or down, and two buttons to move the gripper to predefined positions, one when it is desired to leave the gripper in a store position and the other one when it is desired to operate with the gripper. Figure 6.12 shows two different situations on the gripper panel, when the gripper has no object between its paddles, and when the gripper detects an object between them.



(a) Gripper without object



(b) Griper with object

Figure 6.12: Possibilities on gripper display

Chapter 7

Conclusions

Final conclusions and some lines for future work are discussed here.

7.1 Summary of the thesis and conclusions

This thesis has considered the problem of having several robots working together in order to perform common tasks, that is, cooperate among themselves in the resolution of a task. One of the most important points in the creation of a cooperative system is the definition of the architecture which will give support to that cooperation. In this way, the Acromovi architecture has been developed to perform cooperative tasks with a team of heterogeneous robots.

The Acromovi architecture is basically an agent-based distributed architecture for the programming and controlling of teams of mobile robots. This architecture permits code re-use by using native components, providing the programmer with a set of ready-to-use tested and efficient agents. Also, it allows the scalability of the system and implements the resources sharing, that means that the elements of one robot can be easily accessed by all the other robots of the team.

In order to overcome some problems present on the Acromovi architecture, two new interaction protocols were defined. The first one solves the problem that appears when one agent needs a continuous flow of data from other agent. The second one solves one of the most important problems in distributed systems, the concurrency problem in the access to a certain resource, insuring that only one agent can access to a certain element at each time. This second protocol also insures the physical integrity of the robot, avoiding any action which may put the robot in danger.

Moreover, it has been demonstrated that the architecture may be expanded without difficulty. New agents were implemented to make it possible to integrate a manipulator arm on a mobile base to create a mobile manipulator. These agents facilitate the use of the arm and all its accessories. Thanks to the Acromovi architecture, it is possible to establish the necessary cooperation between the arm and the mobile base in order to accomplish tasks for mobile manipulators.

In order to validate the architecture, new cooperative tasks were implemented. The first one is an application where a group of robots must cooperate in order to safely navigate through an unknown environment. Only one robot has a camera mounted on it and the rest of robots do not have any type of sensors. The robot with the camera calculates the optical flow values from the images that it gets from the camera, and from these values calculates the “time to contact” values that determine the time that the robot has before colliding with an obstacle in front of it. This information is shared among the team in order that any robot can navigate without colliding with the obstacles.

The second cooperative task consists of making it possible for the team of robots to create a certain formation and navigate maintaining this formation. The group is formed of heterogeneous robots where one robot is equipped with a laser system, another one with a camera, and finally, another two robots without any type of sensor. The application consists of two parts or stages. The first one is the creation of the formation, where the robot with the camera can detect the rest of the robots in the environment and indicates to them which is their initial position in the formation. In the second stage, once the formation has been created, the robots must be able to navigate through an environment following the path that the robot with the laser indicates.

The results of the two cooperative tasks have demonstrated that the Acromovi architecture is able to give support to any task which requires cooperation among the robots. In addition, the architecture gives support for all those teams of robots with limited sensory power so that they can perform tasks by means of the sharing of the resources among the team, providing the robots with less sensory power with the capability to use the resources of other robots for their own benefit.

Finally, in an attempt to facilitate the access to the robots of the team and the information that their accessories serve, a system for the teleoperation of the team has been implemented. This system is used for teaching robotics or to facilitate the tasks of programming and debugging in research tasks.

Some of the main contributions of this thesis include: (1) the design and implementation of a control agent-based architecture for the cooperation of a team of heterogeneous mobile robots, (2) the implementation of a new cooperative task for the navigation between several robots by means of the use of the optical flow and time to contact techniques, (3) the development of a new approach in the creation and maintenance of multirobot formations where the robots must cooperate in order to follow the path indicated by the conductor robot without losing the formation shape, (4) a new method for the visual localization of the robots using a common and simple target, (5) an improvement in the calculation of the path to follow by the formation performed by the software of the robots, (6) a new design of formation navigation based on a "conductor-reference" approach with a series of virtual points at a distance from the conductor position for the construction of the formation and Bezier curves for the movement of each robot, (7) the implementation of a cooperative method in order to eliminate the odometry errors of the robots and exploit the resources of each robot to the benefit of the whole team to maintain of the formation while the team is navigating, and finally, (7) the implementation of a teleoperated system to create a simple access to all the robots of the team, the learning of the robots operation, and it is also an important tool to detect failures in the development of cooperative tasks.

7.2 Publications

The development of this thesis has led to the generation of a number of publications in international conferences, some chapters in books and some papers in international journals. The publications associated with this thesis are detailed:

Papers in international journals

- **Control architectures for multirobot teams.**
  - "An integrated agent-based software architecture for mobile and manipulator systems". In Robotica (Special Issue in Mobile Manipulators: basic techniques, new trends and applications), 2007. Authors: P. Nebot and E. Cervera.

  - "Acromovi: an agent-based architecture for cooperative mobile robots". In Industrial Robot: an International Journal (Special Issue on Mobile Robots), in review process. Authors: P. Nebot and E. Cervera.

- **Tele-operation, tele-manipulation and tele-laboratories.**

  - "A Multimodal Interface to Control a Robot Arm via Web: A Case Study on Remote Programming". In IEEE Transactions on Industrial Electronics Journal (Special Section on Human-Robot Interface), 2005. Authors: R. Marín, P. J. Sanz, P. Nebot, R. Wirz.

Chapters in international books

- **Control architectures for multirobot teams.**

  - "Acromovi Architecture: A Framework for the Development of Multirobot Applications". In Mobile Robots: Moving Intelligence. Advanced Robotic Systems, 2006. Authors: P. Nebot, E. Cervera.

- **Tele-operation, tele-manipulation and tele-laboratories.**

  - "TeleLab I: Telelab on Agent-Based Mobile Robots". In Methodology and Tools in Telemanipulation Systems via Internet. Publicacions de la Universitat Jaime I, 2004. Authors: P. Nebot, E. Cervera.
  - "Laboratory on Agent-Based Mobile Manipulation". In Mobile Manipulators: Basic Techniques, New Trends & Applications. Publicacions de la Universitat Jaime I, 2006. Authors: P. Nebot, E. Cervera.

Papers in international conferences

- **Control architectures for multirobot teams.**

  - IEEE Conference on Systems, Man & Cybernetics (SMC 2003) (Washington D.C., USA, 2003). "Agents for Cooperative Heterogeneous Mobile Robotics: a Case Study". Authors: P. Nebot, D. Gómez, E. Cervera.
  - IEEE International Conference on Robotics and Automation (ICRA 2005) (Barcelona, España, 2005). "Agent-based Application Framework for Multiple Mobile Robots Cooperation". Authors: P. Nebot, E. Cervera.
  - 12th International Conference on Advanced Robotics (ICAR 2005) (Seattle, WA, USA, 2005). "A Framework for the Development of Cooperative Robotic Applications". Authors: P. Nebot, E. Cervera.
  - 36th International Symposium on Robotics (ISR 2005) (Tokyo, Japan, 2005). "Acromovi Architecture: Implementation of the Upper Layer". Authors: P. Nebot, E. Cervera.

- **Mobile manipulators.**

  - IEEE Conference on Systems, Man & Cybernetics (SMC 2004) (The Hague, The Netherlands, 2004). "Agent-Based Software Integration for a Mobile Manipulator". Authors: P. Nebot, G. Saintluc, B. Berton, E. Cervera.

- **Optical flow navigation.**
  - 15th International Symposium on Measurement & Control in Robotics (ISMCR 2005) (Brussels, Belgium, 2005). “Distributed Optical Flow Navigation using Acromovi Architecture”. Authors: P. Nebot, E. Cervera.
  - 3rd International Conference on Informatics in Control, Automation and Robotics (ICINCO 2006) (Setúbal, Portugal, 2006). “Optical Flow Navigation over Acromovi Architecture”. Authors: P. Nebot, E. Cervera.
- **Tele-operation, tele-manipulation and tele-laboratories.**
  - IEEE Conference on Systems, Man & Cybernetics (SMC 2003) (Washington D.C., USA, 2003). “Multirobot Internet-Based Architecture for Telemanipulation: Experimental Validation”. Authors: R. Marín, P. J. Sanz, P. Nebot, R. Esteller.
  - IEEE Conference on Systems, Man & Cybernetics (SMC 2003) (Washington D.C., USA, 2003). “The Internet-Based UJI Tele-Lab: System Architecture”. Authors: R. Marín, P. J. Sanz, P. Nebot, R. Esteller.
  - 1st IFAC Symposium on Telematics Applications in Automation and Robotics (TA 2004) (Helsinki, Finland, 2004). “Multirobot System Architecture and Performance Issues for the UJI Robotics TeleLab”. Authors: R. Marín, P. J. Sanz, P. Nebot, R. Esteller, R. Wirz.

Papers in national conferences

- **Control architectures for multirobot teams.**
  - X Conferencia de la Asociación Española de Inteligencia Artificial (CAEPIA 2003) (San Sebastián, 2003). “Arquitectura Distribuida para un Equipo de Robots Móviles Heterogéneos” (in Spanish). Authors: P. Nebot, D. Gómez, E. Cervera.
  - Campus Multidisciplinar en Percepción e Inteligencia (CMPI 2006) (Albacete, 2006). “La Arquitectura Acromovi: una Arquitectura para Tareas Cooperativas de Robots Móviles” (in Spanish). Authors: P. Nebot, E. Cervera.

7.3 Future work

Work on the architecture and the development of cooperative tasks has opened several lines that could be considered for future work:

- **Architecture for cooperative tasks.** The architecture proposed in this thesis has been designed for the development and implementation of cooperative tasks. This architecture includes two new interaction protocols. With the approach followed in this protocols, when one agent wins the access to a certain element, it has this access until it finishes its task. An important improvement could be to introduce a maximum of time for doing the tasks. Once this time is finished, the agent must wait until getting permission once more. Or another possibility could be to add a system of priorities, thus, if an agent has an urgent task to do it can get access temporarily, throwing out the agents that in that moment are using the element, to use the element immediately and does not have to wait to get permission. Or even, a mixture of the two techniques would be more profitable.

- **Cooperative navigation using optical flow.**
  - One of the problems with the optical flow technique is that it is impossible to distinguish those situations when an object is too close to the robot. Sonars can be used to distinguish those situations and allow the robot to perform the evasive maneouvre.
  - Another important problem relies on the calculation of the optical flow in homogeneous surfaces. New techniques or ways to detect this situation must be studied and improve the system in order to deal with it.
  - Also, it is important to try of reducing the time consumption by the calculation of the optical flow. For that, new methods which are faster than the used in this work can be used. One possible option that implies a little variation compared to the original method is described in [350]. So, with a few changes it is possible to get a faster system and capable of reacting in a more reliable way to changes in the environment.
- **Cooperative formations of multiple mobile robots.**
  - The localization of the robots by means of the colored targets has been a hazardous work due to the sensitivity of the vision system to the lighting conditions. Work could be done on the development of more robust localization procedures using a combination of different features to detect the target in the image space.
  - One of the points not yet developed is that concerning how to change the shape of the formation while the robots are moving. This can be useful in order to avoid obstacles or to pass through narrow passages where the formation does not fit. With the actual implementation of the system it is easy to perform this task as long as the robots can stop and reshape the formation. To perform this task with the formation in movement, the Bezier curves can be used to move the robots to virtual points that determinate the position that they must occupy in the formation after the reshaping.
  - Another point not developed is the detection of obstacles in the path of the formation. All the static obstacles are detected and avoided by means of the generation of the path that the robots must follow by the navigation system, but at this moment the dynamic obstacles can not be detected. It is necessary to study how it might be possible, with the available sensors, to detect dynamic obstacles and avoid them. It is also important to study how to avoid the obstacle, that is, if it is necessary to reshape the formation or if it is possible to move the formation in block.
- **Teleoperation of a team of mobile robots.** It would be profitable to improve the teleoperated system including techniques to give the user an improved perception of the environment. For this, it would be necessary to introduce techniques creating a virtual representation of the environment in 3D which could be implemented. Moreover, it would be necessary to include new ways to introduce orders for the robots on a higher level. And finally, while at the moment the orders can only be performed in one robot at a time, it is therefore necessary to introduce a mechanism to facilitate the specification of orders for all the team in conjunction.

Bibliography

[1] E.U. Acar, H. Choset, Y. Zhang, and M.S. Schervish. Path Planning for Robotic Deminig: Robust Sensor-Based Coverage of Unstructured Enviroments and Probabilistic Methods. *International Journal of Robotic Research*, 22(7-8):441–466, 2003.

[2] J. Adams. Human Factors Experimental Analysis of the MASC System. Technical Report MSCIS-96-11, University of Pennsylvania, Philadelphia, Pennsylvaniaa, 1996.

[3] G. Adiv. Determining Three-Dimensioal Motion and Strucuture from Optical Flow Generated by Several Moving Objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7(4):384–401, 1985.

[4] R. Alami, S. Fleury, M. Herrb, F. Ingrand, and F. Robert. Multi-Robot Cooperation in the MARTHA Project. *IEEE Robotics & Automation Magazine*, 5(1):36–47, 1998.

[5] J.S. Albus. Hierarchical Interaction between Sensory Processing and World Modeling in Intelligent Systems. In *Proceedings of the 5th IEEE International Symposium on Intelligent Control*, pages 866–875, 1990.

[6] J.Y. Aloimonos, I. Weiss, and A. Bandopadhay. Active Vision. *International Journal on Computer Vision*, pages 333–356, 1987.

[7] Y. Aloimonos. Introduction: Active Vision Revisited. In *Active Perception, volume I of Advances in Computer Science Series*. Lawrence Erlbaum Associates, 1993.

[8] W. Amai, J. Fahrenholtz, and C. Leger. Hands-Free Operation of a Small Mobile Robot. *Autonomous Robots*, 11(1):69–76, 2001.

[9] P. Anandan. A Computational Framework and an Algorithm for Measurement of Visual Motion. *International Journal of Computer Vision*, 2:283–310, 1989.

[10] N. Ancona and T. Poggio. Optical Flow from 1d Correlation: Application to a Simple Time-to-Crash Detector. In *Proceedings of the International Conference on Computer Vision*, 2005.

[11] T. L. Anderson, L. Hendriksen, and O. Raven. Animal Behaviour as a Paradigm for Developing Robot Autonomy. *Designing Autonomous Agents*, pages 145–168, 1991.

[12] T. Arai, H. Ogata, and T. Suzuki. Collision Avoidance among Multiple Robots using Virtual Impedance. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 479–485, 1989.

[13] T. Arai, E. Pagello, and L. E. Parker. Guest Editorial: Advances in Multirobot Systems. *IEEE Transactions on Robotics and Automation*, 18(5):655–661, 2002.

[14] M.A. Arbib. Perceptual structures and distributed motor control. In *Handbook of Physiology: Motor Control*, pages 809–813. The MIT Press, 1981.

[15] M.A. Arbib. Schema Theory. In *The Encyclopedia of Artificial Intelligence*, pages 1427–1443. Wiley-Interscience, 1992.

[16] R.C. Arkin. Path Planning for a Vision-Based Autonomous Robot. In *Proceedings of the SPIE Conference on Mobile Robots*, pages 240–249, 1986.

[17] R.C. Arkin. Motor Schema Based Navigation for a Mobile Robot: An Approach to Programming by Behavior. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 264–271, 1987.

[18] R.C. Arkin. Neuroscience in Motion: The Application of Schema Theory to Mobile Robotics. In *Visuomotor Coordination: Amphibians, Comparisons, Models and Robots*, pages 649–672. Plenum Press, 1989.

[19] R.C. Arkin. Cooperation without Comunication: Multiagent Schema Based Robot Navigation. *Journal of Robotics Systems*, 9(3):351–369, 1992.

[20] R.C. Arkin. Modeling Neural Function at the Schema Level: Implications and Results for Robotic Control. In *Biological Neural Networks in Invertebrate Neurology and Robotics*, pages 383–410. Academic Press, 1993.

[21] R.C. Arkin. *Behavior-Based Robotics*. Intelligent Robotics and Autonomous Agents series. The MIT Press, 1998.

[22] H. Asama, A. Matsumoto, and Y. Ishida. Design of an Autonomous and Distributed Robot System: ACTRESS. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 283–290, 1989.

[23] R. Atienza and A. Zelinsky. A Practical Zoom Camera Calibration Technique: an Application of Active Vision for Human-Robot Interaction. In *Proceedings of the Australian Conference on Robotics and Automation*, pages 85—-90, 2001.

[24] Y. Baba, T. Kurita, M. Tanaka, S. Akaho, S. Umeyama, and T. Mishima. An Experimental Foveated Vision System using Fish-Eye Lens and Integration of Retinal Images. In *Proceedings of the International Technical Conference on Circuits/Systems, Computers and Communications*, pages 1–8, 1999.

[25] P. Backes, G. Tharp, and K. Tso. TheWeb Interface for Telescience (WITS). In *Proceedings of the IEEE International Conference on Robotics and Automation*, 1997.

[26] R. Bajcsy. Active Perception. *IEEE Proceedings (Special Issue on Computer Vision)*, 76(8):996–1006, 1988.

[27] T. Balch and R.C. Arkin. Communication in Reactive Multiagent Robotic Systems. *Autonomous Robots*, 1(1):27–52, 1994.

[28] T. Balch and R.C. Arkin. Motor Schema-Based Formation Control for Multiagent Robot Teams. In *Proceedings of the First International Conference on Multi-Agent Systems*, pages 10—-16, 1995.

[29] T. Balch and R.C. Arkin. Behaviour-Based Formation Control for Multiagent Robot Teams. *IEEE Transaction on Robotics and Automation*, 14(6):926–993, 1998.

[30] T. Balch and R.C. Arkin. Behavior-based Formation Control for Multi-robot Teams. *IEEE Transactions on Robotics and Automation*, 14(6):926–939, 1999.

[31] T. Balch and M. Hybinette. Social Potentials for Scalable Multirobot Formations. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 73–80, 2000.

[32] D.H. Ballard. Animate Vision. *Artificial Intelligence*, 48:57–86, 1996.

[33] P. Ballou. Improving Pilot Dexterity with a Telepresent ROV. In *Proceedings of the International Conference on Robotics and Automation*, 2000.

[34] S.T. Barnard and W.B. Thompson. Disparity Analysis of Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(4):333–340, 1980.

[35] J.L. Barron, D.J. Fleet, and S.S. Beauchemin. Performance of Optical Flow Techniques. *Internation Journal of Computer Vision*, 12(1):43–77, 1994.

[36] J.L. Barron and A. Liptay. Optic Flow to Measure Minute Increments in Plant Growth. *Bioimaging*, 2(1):57–61, 1994.

[37] A. Basu. Active Calibration of Cameras: Theory and Implementation. *IEEE Transactions on Systems, Man and Cybernetics*, 25(2):256–265, 1995.

[38] B. Bayle, J.Y. Fourquet, and M. Renaud. Manipulability of Wheeled Mobile Manipulators: Application to Motion Generation. *International Journal of Robotics Research*, 22(7):565–581, 2003.

[39] J.D. Bayliss, C.M. Brown, R.L. Carceroni, C. Eveland, C. Harman, A. Singhal, and M. Van Wie. Mobile Robotics 1997. Technical Report TR661, Department of Computer Science, University of Rochester, 1997.

[40] M. Bayouth and C. Thorpe. An AHS Concept Based on an Autonomous Vehicle Architecture. In *Proceedings of the 3rd Annual World Congress on Intelligent Transportation Systems*, 1996.

[41] S.S. Beauchemin and J.L. Barron. The Computation of Optical Flow. *ACM Computing Surveys*, 27(3):433–467, 1995.

[42] C. Becker, H. Gonzalez-Baños, J.C. Latombe, and C. Tomas. An Intelligent Observer. *Lecture Notes in Control and Information Sciences*, 223:153–160, 1995.

[43] F. Bellifemine, G. Caire, A. Poggi, and G. Rimassa. JADE A White Paper. *EXP in Search of Innovation (Special Issue on JADE)*, 3(3):6–19, 2003.

[44] E. Ben-Jacob, Y. Aharonov, and Y. Shapira. Bacteria Harnessing Complexity. *Biofilms*, 1:239–263, 2004.

[45] M. Benda, V. Jagannathan, and R. Dodhiawalla. On Optimal Cooperation of Knowledge Sources. Technical Report BCS-G2010-2, Boeing Advanced Technology Center, 1985.

[46] G. Beni. The Concept of a Cellular Robot. In *Proceedings of the 3rd IEEE Symposium on Intelligent Control*, pages 57–61, 1988.

[47] S.M. Benoits and F.P. Ferrie. Monocular Optical Flow for Real-Time Vision Systems. Technical report, Center for Intelligent Machines, McGill University, 1996.

[48] C. Bererton, L.E. Navarro-Serment, R. Grabowski, C. Paredis, and P. Khosla. Millibots: Small Distributed Robots for Surveillance and Mapping. In *Proceeding of the Government Microcircuit Applications Conference*, 2000.

[49] M.J. Black and P. Anandan. A Model for the Detection of Motion over Time. In *Proceedings of the International Conference on Computer Vision*, pages 33–37, 1990.

[50] T. Blackmon, S. Thayer, J. Teza, V. Broz, J. Osborn, and M. Hebert. Virtual Reality Mapping System for Chernobyl Accident Site Assessmen. In *Proceedings of the SPIE*, volume 3644, pages 338–345, 1999.

[51] H. Bojinov, A. Casal, and T. Hogg. Emergent Structures in Modular Self-Reconfigurable Robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1734–1741, 2000.

[52] A.H. Bond and L. Gasser. An Analysis of Problems and Research in DAI. In *Readings in Distributed Artificial Intelligence*, pages 3–35. Morgan Kaufmann Publishers, 1988.

[53] P. Bouthemy and E. Francois. Motion Segmentation and Qualitative Dynamic Scene Analysis from an Image Sequence. *International Journal of Computer Vision*, 10(2):159–182, 1993.

[54] D.C. Brogan and J.K. Hodgins. Group Behaviors for Systems with Significant Dynamics. *Autonomous Robots*, 4(1):137–153, 1997.

[55] A. Brooks, S. Abdallah, A. Zelinsky, and J. Kieffer. A Multi-Modal Approach to Real-Time Active Vision. In *Proceedings of the International Conference on Intelligent Robotics*, pages 527–532, 1998.

[56] R. A. Brooks. Intelligence without Reason. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pages 569–590, 1991.

[57] R.A. Brooks. A Robust Layered Control System for a Mobile Robot. *IEEE Journal of Robotics and Automation*, RA-2:14–23, 1986.

[58] R.A. Brooks. A Robot that Walks; Emergent Behaviours from a Carefully Evolved Network. *Neural Computation*, 81(2), 1989.

[59] R.A. Brooks. Intelligent without Representation. *Artificial Intelligence*, 47(1-3):139–160, 1991.

[60] R.A. Brooks. New Approaches to Robotics. *Science*, 253:1227–1232, 1991.

[61] C. Brown. Prediction and Cooperation in Gaze Control. *Biological Cybernetics*, 63(1):61–70, 1990.

[62] D. Brugali, M. Alencastre-Miranda, L. Muñoz-Gomez, D. Botturi, and L. Cragg. Trends in Software Environments for Networked Robotics. In D. Brugali, editor, *Software Engineering for Experimental Robotics*, volume 30 of *Springer Tracts in Advanced Robotics*, pages 401–408. Springer, 2007.

[63] J. Buhmann, W. Burgard, A.B. Cremers, D. Fox, T. Hofmann, F. Schneider, J. Strikos, and S. Thrun. The Mobile Robot RHINO. *AI Magazine*, 16(2):31–38, 1995.

[64] W. Burgard, M. Moors, D. Fox, R. Simmons, and S. Thrun. Collaborative Multirobot Exploration. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 476–481, 2000.

[65] W. Burgard, M. Moors, C. Stachniss, and F. Schneider. Coordinated Multirobot Exploration. *IEEE Transactions on Robotics*, 1(3):376–385, 2005.

[66] P. Burlina and Chellapa R. Time-to-x: Analysis of Motion through Temporal Parameters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 461–468, 1994.

[67] P.J. Burt. Fast Filter Transforms for Image Processing. *Computer Graphics and Image Processing*, 16:20–51, 1981.

[68] P. Caloud, W. Choi, J.C. Latombe, C. Le Pape, and M. Yim. Indoor Automation with many Mobile Robots. In *Proceeedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 67–72, 1990.

[69] T. Camus. Real-Time Quantized Optical Flow. In *Proceedings of the IEEE Conference on Computer Architecture for Machine Perception*, pages 126–131, 1995.

[70] Y.U. Cao, A.S. Fukunaga, and A.B. Kahng. Cooperative Mobile Robotics: Antecedents and Directions. *Autonomous Robots*, 4:1–23, 1997.

[71] A. Cardenas, B. Goodwine, S. Skaar, and M. Seelinger. Vision-Based Control of a Mobile Base and On-Board Arm. *International Journal of Robotics Research*, 22(7):677–698, 2003.

[72] R. Carelli, C. Soria, and O. Nasisi. Stable AGV Corridor Navigation with Fused Vision-Based Control Signals. In *Proceedings of the 28th Annual Conference of IEEE Industry Electronics Society*, pages 2433—-2438, 2002.

[73] B. Carpentieri and J.A. Storer. A Split-Merge Parallel Block-Matching Algorithm for Video Displacement Estimation. In *Proceedings of the Data Compression Conference*, pages 239–248, 1992.

[74] A. Castano, R. Chokkalingam, and P. Will. Autonomous and Selfsufficient Conro Modules for Reconfigurable Robots. *Distributed Autonomous Robotic Systems*, 4:155–164, 2000.

[75] A. Castellani, S. Galvan, D. Botturi, and P. Fiorini. Advanced Teleoperation Architecture. In D. Brugali, editor, *Software Engineering for Experimental Robotics*, volume 30 of *Springer Tracts in Advanced Robotics*, pages 409–430. Springer, 2007.

[76] D.A. Cáceres, H. Martínez, M.A. Zamora, and L.M.T. Balibrea. A Real-Time Framework for Robotics Software. In *Proceeedings of the International Conference on Computer Integrated Manufacturing*, 2003.

[77] Q. Chen and J. Y. S. Luh. Coordination and Control of a Group of Small Mobile Robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2315–2320, 1994.

[78] D. L. Cheney and R. M. Seyfarth. *How Monkeys See the World*. University of Chicago Press, 1990.

[79] S.Y. Chiem and E. Cervera. Vision-based Robot Formations with Bézier Trajectories. In *Proceeedings of the 8th Conference on Intelligent Autonomous System*, 2004.

[80] X. Clady, F. Collange, F. Jurie, and P. Martinet. Objet Tracking with a Pan Tilt Zoom Camera, Application to Car Driving Assistance. In *Proceedings of the International Conference on Advanced Robotics*, pages 1653—-1658, 2001.

[81] J. Connell. The Omni Photovore: How to Build a Robot that Thinks like a Roach. *Omni Magazine*, 1988.

[82] J. Connell. A Behavior-Based Arm Controller. *IEEE Transactions on Robotics and Automation*, 5(6):784–791, 1989.

[83] J. Connell. A Colony Architecture for an Artificial Creature. Technical Report 1151, Massachusetts Institute of Technology, AI Laboratory, 1989.

[84] J. Connell. Minimalist Mobile Robotics: A Colony-Style Architecture for an Artificial Creature. *Academic Press*, 1990.

[85] J. Connell. Design your Own Robots. *Popular Electronics*, 1991.

[86] D. Coombs and C. Brown. Real-time Binocular Smooth Pursuit. *International Journal of Computer Vision*, 11(2):147–164, 1993.

[87] D. Coombs, M. Herman, T.H. Hong, and M. Nashman. Real-Time Obstacle Avoidance using Central Flow Divergence and Peripheral Flow. *IEEE Transactions on Robotics and Automation*, 14(1):49–59, 1998.

[88] B. Cooper. Driving on the Surface of Mars Using the Rover Control Workstation. In *Proceedings of the 5th International Symposium on Space Mission Operations and Ground Data Systems*, 1998.

[89] V. Cornilleau-Peres and J. Droulez. Stereo Correspondence from Optical Flow. In *Proceedings of the European Conference on Computer Vision*, pages 326–330, 1990.

[90] I.J. Cox and G.T. Wilfong. *Autonomous Robot Vehicles*. Springer Verlag, 1990.

[91] B. J. Crespi and J. C. Choe. *The Evolution of Social Behaviour in Insects and Arachnids*. Cambridge University Press, 1997.

[92] G.D. Cubber, S. Berrabah, and H. Sahli. A Bayesian Approach for Color Consistency based Visual Servoing. In *Proceedings of the International Conference on Advanced Robotics*, pages 983—-990, 2003.

[93] A. Das, R. Fierro, V. Kumar, B. Southall, J. Spletzer, and C. Taylor. Real-Time Vision-Based Control of a Nonholonomic Mobile Robot. In *Proceedings of the International Conference on Robotics and Automation*, pages 1714—-1719, 2001.

[94] K. Das, R. Fierro, V. Kumar, J. P. Ostrowski, J. Spletzer, and C.J. Taylor. A Vision-Based Formation Control Framework. *IEEE Transactions on Robotics and Automation*, 18(5):813–825, 2002.

[95] T. Deacon. *The Symbolic Species: The Co-evolution of Language and the Human Brain*. Penguin Books, 1997.

[96] T. Dean and M. Wellman. *Planning and Control*. Morgan-Kaufmann, 1991.

[97] G. Dedeoglu and G. Sukhatme. Landmark-Based Matching Algorithm for Cooperative Mapping by Autonomous Robots. *Distributed Autonomous Robotic Systems*, 4:251–260, 2000.

[98] J. Deneubourg, S. Goss, and G. Sandini. Self-Organizing Collection and Transport of Objects in Unpredictable Environments. In *Proceedings of the Japan-USA Symposium on Flexible Automation*, pages 1093–1098, 1990.

[99] J.D. Desai, V. Kumar, and J.P. Ostrowski. Control of Changes in Formation for a Team of Mobile Robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1556–1561, 1999.

[100] J.P. Desai, J. Ostrowski, and V. Kumar. Modeling and Control of Formations of Nonholonomic Mobile Robots. *IEEE Transactions on Robotics and Automation*, 17(6):905–908, 2001.

[101] A. Dev, B.J.A. Krose, and F.C.A. Groen. Navigation of a Mobile Robot on the Temporal Development of the Optic Flow. In *Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems*, pages 558—-563, 1997.

[102] M. Draper and H. Ruff. Multi-Sensory Displays and Visualization Techniques Supporting the Control of Unmanned Air Vehicles. In *Proceedings of the Workshop on Interactive Robotics and Entertainment*, 2000.

[103] A. Drogoul and J. Ferber. From Tom Thumb to the Dockers: some Experiments with Foraging Robots. In *Proceedings of the 2nd International Conference on Simulation of Adaptive Behavior*, pages 451–459, 1992.

[104] E. Dubois. The Sampling and Reconstruction of Time-Varying Imagery with Application in Video Systems. *Proceedings of the IEEE*, 73(4):502–522, 1985.

[105] G. Dudek, M. Jenkin, and E. Milios. A Taxonomy of Multirobot Systems. In *Robot Teams: From Diversity to Polymorphism*, pages 3—-22. AK Peters, 2002.

[106] J.H. Duncan and T. Chou. On the Detction of Motion and the Computation of Optical Flow. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(3):346–352, 1992.

[107] J.O. Eklundh and K.Pahlavan. Heads, Eyes and Head-Eye Systems. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(1):33–49, 1993.

[108] J.O. Eklundh, P. Nordlund, and T. Uhlin. Issues in Active Vision: Attention and Cue Integration / Selection. In *Proceedings of the British Machine Vision Conference*, pages 1–12, 1996.

[109] W. Enkelmann. Obstacle Detection by Evaluation of Otical Flow Fields from Image Sequences. In *Proceedings of the European Conference on Computer Vision*, pages 134–138, 1990.

[110] M.S. Erden, K. Leblebicioglu, and U. Halici. Multi-Agent System-Based Fuzzy Controller Design with Genetic Tuning for a Mobile Manipulator Robot in the Hand Over Task. *Journal of Intelligent and Robotic Systems*, 39(3):287–306, 2004.

[111] P.E. Rybski et al. Performance of a Distributed Robotic System using Shared Communication Channels. *IEEE Transactions on Robotics and Automation*, 22(5):713–727, 2002.

[112] R. Siegwart et al. A Large Scale Installation of Personal Robots. *Robotics and Autonomous Systems*, 42(3-4):203–222, 2003.

[113] S. Bahadori et al. Autonomous System for Search and Rescue. In *Rescue Robotics*. Springer-Verlag, 2005.

[114] J. Ferber and O. Gutknecht. A Meta-model for the Analisys and Design of Organizations in Multi-agent Systems. In *Proceeedings of the 3rd International Conference on Multi-agent Systems*, pages 128–135, 1998.

[115] I. Fermin and A. Imiya. Two-Dimensional Motion Computation by Randomized Method. Technical Report ICS-4-6-1994, Department of Information and Computer Sciences, Chiba University, 1994.

[116] C. Fermüller and Y. Aloimonos. Tracking Facilitates 3D Motion Estimation. *Biological Cybernetics*, 67:259–268, 1992.

[117] C. Fermüller and Y. Aloimonos. Qualitative Egomotion. *Internation Journal of Computer Vision*, 15:7–29, 1995.

[118] W. Ferrel and T. Sheridan. Supervisory Control of Remote Manipulation. *IEEE Spectrum*, 4(10):81–88, 1967.

[119] R. Fierro, A.K. Das, V. Kumar, and J.P. Ostrowski. Hybrid Control of Formations of Robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 157–162, 2001.

[120] R. Fierro, P. Song, A.K. Das, and V. Kumar. Cooperative Control of Robot Formations. In *Cooperative Control and Optimization Series*. Kluwer, 2001.

[121] R.J. Firby, P.N. Prokopowicz, and M.J. Swain. The Animate Agent Architecture. In *Artificial Intelligence and Mobile Robots*, pages 243–275. The MIT Press, 1998.

[122] D.J. Fleet and A.D. Jepson. Computation of Component Image Velocity from Local Phase Information. *International Journal of Computer Vision*, 5(1):77–104, 1990.

[123] D.J. Fleet and K. Langley. Recursive Filters for Optical Flow. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(1):61–67, 1995.

[124] T. Fong, F. Conti, S. Grange, and C. Baur. Novel Interfaces for Remote Driving: Gesture, Haptic, and PDA. In *Proceedings of the SPIE Telemanipulator and Telepresence Technologies VII*, 2000.

[125] T. Fong, H. Pangels, D. Wettergreen, E. Nygren, B. Hine, P. Hontalas, and C. Fedor. Operator Interfaces and Network-Based Participation for Dante II. In *Proceedings of the 25th International Conference on Environmental Systems*, 1995.

[126] T. Fong and C. Thorpe. Robot as Partner: Vehicle Teleoperation with Collaborative Control. In *Proceedings of the NRL Workshop on Multi-Robot Systems*, 2002.

[127] T. Fong, C. Thorpe, and C. Baur. Collaboration, Dialogue, and Human-Robot Interaction. In *Proceedings of the 10th International Symposium of Robotics Research*, 2001.

[128] G.L. Foresti and B. Pani. Monitoring Motorway Infrastrucutres for Detection of Dangerous Events. In *Proceedings of the International Conference on Image Analysis and Processing*, 1999.

[129] D. Fox, W. Burgard, H. Kruppa, and S. Thrun. A Probabilistic Approach to Collaborative Multi-Robot Localization. *Autonomous Robots*, 8(3), 2000.

[130] D. Foyle. Proposed Evaluation Framework for Assessing Operator Performance with Multisensor Displays. In *Proceedings of the SPIE*, volume 1666, pages 514–525, 1992.

[131] J. Fredslund and M.J. Mataric. A General, Local Algorithm for Robot Formationss. *IEEE Transactions on Robotics and Automation (Special Issue on Advances in Multi-Robot Systems)*, 18(5):837–846, 2002.

[132] J. Freslund and M.J. Mataric. A General, Local Algorithm for Robot Formations. *IEEE Transactions on Robotics and Automation*, 2001.

[133] M. Fujii, T. Yamamoto, and T. Fujinami. Stable Formation Driving of Mobile Robots with Hybrid Strategy. In *Dynamic Systems Approach for Embodiment and Sociality*, pages 238–242. Springer, 2003.

[134] T. Fukuda and G. Iritani. Construction Mechanism of Group Behavior with Cooperation. In *Proceeedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 535–542, 1995.

[135] T. Fukuda, H. Mizoguchi, K. Sekiyama, and F. Arai. Group Behaviour Control for MARS (Micro Autonomous Robotic Systems. In *Proceedings of the 1999 IEEE International Conference on Robotics & Automation*, pages 1550–1555, 1999.

[136] T. Fukuda, S. Nadagawa, Y. Kawauchi, and M. Buss. Structure Decission for Self Organizing Robots Based on Cell Structures - Cebot. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 695–700, 1989.

[137] T. Fukuda and S. Nakagawa. A Dynamically Reconfigurable Robotic System (Concept of a System and Optimal Configurations). In *Proceedings of the IEEE International Conference on Industrial Electronics, Control and Instrumentation*, pages 588–595, 1987.

[138] D.W. Gage. Command Control for Many-Robot Systems. *Unmanned Systems Magazine*, 10(4):28–34, 1992.

[139] C. Galindo, J. Gonzalez, and J.A. Fernandez-Madrigal. A Control Architecture for Human-Robot Integration: Application to a Robotic Wheelchair. *IEEE System, Man and Cybernetics Part B*, 36(6), 2006.

[140] B. Galvin, B. McCane, K. Novins, D. Mason, and S. Mills. Recovering Motion Fields: An Evaluation of Eight Optical Flow Algorithms. In *Proceedings of the British Machine Vision Conference*, pages 195–204, 1998.

[141] D.F.T. Gamarra, M. Sarcinelli-Filho, and T.F. Bastos-Filho. Controlling the Navigation of a Mobile Robot in a Corridor with Redundant Controllers. In *Proceedings of the International Conference on Robotics and Automation*, pages 3855—-3860, 2005.

[142] E. Gat. ALFA: A Language for Programming Reactive Robotic Control Systems. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1116–1120, 1991.

[143] B.P. Gerkey, R.T. Vaughan, and A. Howard. The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems. In *Proceedings of the International Conference on Advanced Robotics*, pages 317–323, 2003.

[144] A. Giachetti, M. Campani, and V. Torre. The Use of Optical Flow for Autonomous Navigation. In *Proceedings of the European Conference on Computer Vision*, pages 146–151, 1994.

[145] M.C.A. Giachetti and V. Torre. The Use of Optical Flow for Road Navigation. *IEEE Transactions on Robotics and Automation*, 14:34–48, 1998.

[146] M. Glumm, P. Kilduff, and A. Masley. A Study on the Effects of Lens Focal Length on Remote Driver Performance. Technical Report ARL-TR-25, Army Research Laboratory, Aldephi, Maryland, 1992.

[147] J. Goetz and S. Kiesler. Cooperation with a Robotic Assistant. In *Proceedings of the Conference on Human Factors in Computing Systems*, 2002.

[148] R. Gordon and S. Talley. *Essential JMF: Java Media Framework*. Prentice Hall, 1998.

[149] S. Grand and D. Cliff. Creatures: Entertaiment Software Agents with Artificial Life. *Autonomous Agents and Multiagent Systems*, 1(1):39–58, 1998.

[150] S. Grange, T. Fong, and C. Baur. Effective Vehicle Teleoperation on the World Wide Web. In *Proceedings of the International Conference on Robotics and Automation*, 2000.

[151] J. Grefenstette and R. Daley. Methods for Competitive and Cooperative Co-evolution. In *Adaptation, Coevolution and Learning in Multiagent Systems: Papers from the 1996 AAAI Spring Symposium*, pages 45–50, 1996.

[152] R. Grupen and T. Henderson. Autochtonous Behaviors–Mapping Perception to Action. In *Traditional and Non-Traditional Robotic Sensors*, volume F-63 of *NATO ASI Series*, pages 867–871. Springer-Verlag, 1990.

[153] O. Gutknecht and J. Ferber. MadKit Reference page. http://www.madkit.org. Last Visited: 10/09/2007.

[154] D. Hainsworth. Teleoperation User Interfaces for Mining Robotics. *Autonomous Robots*, 11(1):19–28, 2001.

[155] A. Halme, , and J. Suomela. Tele-Existence Techniques of Heavy Work Vehicles. *Autonomous Robots*, 11(1):29–38, 2001.

[156] T. Haynes and S. Sen. Evolving Behavioral Strategies in Predators, and Prey. In *Adaptation and Learning in Multi-Agent Systems*, pages 113–126. Springer, 1986.

[157] D.J. Heeger. Optical Flow using Spatiotemporal Filters. *International Journal of Computer Vision*, 1:279–302, 1988.

[158] F. Heitz and P. Bouthemy. Multimodal Estimation of Discontinuous Optical Flow Using Markov Random Fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(12):1217–1232, 1993.

[159] Y. Hirata, K. Kosuge, H. Asama, H. Kaetsu, and K. Kawabata. Coordinated Transportation of a Single Object by Multiple Mobile Robots Without Position Information of Each Robot. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2024–2029, 2000.

[160] R. Hogg, A. Rankin, S. Roumeliotis, M. Henry, D. Helmick, C. Bergh, and L. Matthies. Algorithms and Sensors for Small Robot Path Following. In *Proceedings of the International Conference on Robotics and Automation*, pages 3850—-3857, 2002.

[161] B.K.P. Horn. *Robot Vision*. MIT Press, 1986.

[162] B.K.P. Horn and B.G. Schunk. Determining Optical Flow. *Artificial Intelligence*, 17:185–201, 1981.

[163] I. Horswill. Polly: A Vision-Based Artificial Agent. In *Proceedings of the 11th National Conference on Artificial Intelligence*, 1993.

[164] K. Hosoda, H. Moriyama, and M. Asada. Visual Servoing Utilizing Zoom Mechanism. In *Proceedings of the International Conference on Advanced Robotics*, pages 178—-183, 1995.

[165] A. Howard. Mezzanine User Manual; Version 0.00. Technical Report IRIS-01-416, USC Robotics Laboratory, University of Sourthern California, 2002.

[166] M.J. Huber and E.H. Durfee. Deciding When to Commit to Action during Observation-Based Coordination. In *Proceedings of the 1st International Conference on Multi-Agent Systems*, pages 163–170, 1995.

[167] L. Hugues. Collective Grounded Representations for Robots. *Distributed Autonomous Robotic Systems*, 4:79–88, 2000.

[168] Y.S. Hung and H.T. Ho. A Kalman Filter Approach to Direct Depth Estimation Incorporating Surface Structure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(6):570–575, 1999.

[169] J. Hynoski and H.R. Wu. Active Vision - a Survey of the Field and Research Directions. Technical Report 95-04, Department of Robotics and Digital Technology, Monash University, May 1995.

[170] C. Isik and A.M. Meystel. Pilot Level of a Hierarchical Controller for an Unmanned Mobile Robot. *IEEE Journal of Robotics and Automation*, 4(3):241–255, 1988.

[171] R.C. Jain. Direct Computation of the Focus of Expansion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(1):58–63, 1983.

[172] R.C. Jain. Segmentation of Frame Sequences Obtained by a Moving Observer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(5):624–629, 1984.

[173] M.R.M. Jenkin, A.D. Jepson, and J.K. Tsotsos. Techniques for Disparity Measurement. *CVGIP: Image Understanding*, 53(1):14–30, 1991.

[174] J. Jennings, G. Whelan, and W. Evans. Cooperative Search and Rescue with a Team of Mobile Robots. In *Proceedings of the IEEE International Conference on Advanced Robotics*, pages 193—-200, 1997.

[175] N.R. Jennings and T. Witting. Archon: Theory and Practice. In *Distributed Artificial Intelligence: Theory and Praxis*, pages 179–195. Kluwer Academic Press, 1992.

[176] J. Johnson and M. Sugisaka. Complexity Science for the Design of Swarm Robot Control Systems. In *Proceedings of the 26th Annual Conference of the IEEE Industrial Electronics Society (IECON)*, pages 695–700, 2000.

[177] J.L. Jones, A.N. Flynn, and B.A. Seiger. *Mobile Robots: Inspiration to Implementation*. A K Peters, 1999.

[178] D. Jung and A. Zelinsky. An Architecture for Distributed Cooperative Planning in a Behaviour-Based Multi-Robot System. *Journal of Robots and Autonomous Systems*, 26(2–3):149–174, 1999.

[179] D. Jung and A. Zelinsky. Grounded Symbolic Communication between Heteronegeous Cooperating Robots. *Autonomous Robots*, 8(3):269–292, 2000.

[180] L. Kaelbling and S. Rosenschein. Action and Planning in Embedded Agents. In *Designing Autonomous Agents*, pages 35–48. The MIT Press, 1991.

[181] D. Kaiser and R. Losick. How and Why Bacteria Talk to Each Other. *Cell*, 73(5):873–885, 1993.

[182] J. Kay. *STRIPE: Remote Driving using Limited Image Data*. PhD thesis, Carnegie Mellon University, 1997.

[183] O. Khatib. Mobile Manipulation: The Robotic Assistant. *Robotics and Autonomous Systems*, 26(2-3):175–183, 1999.

[184] O. Khatib, K. Yokoi, K. Chang, D. Ruspini, R. Holmberg, and A. Casals. Vehicle/Arm Coordination and Multiple Mobile Manipulator Decentralized Cooperation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 546–553, 1996.

[185] K. Konolige. The SRI Small Vision System. http://www.ai.sri.com/ konolige/svs/svm.htm. Last Visited: 10/09/2007.

[186] K. Konolige, K.L. Myers, E.H. Ruspini, and A. Saffiott. The Saphira Architecture: A Design for Autonomyn. *Journal of Experimental & Theoretical Artificial Intelligence*, 9(1):215–235, 1997.

[187] P. Kostelnik, M. Samulka, and M. Janosik. Scalable Multi-Robot Formations using Local Sensing and Communication. In *Proceedings of the Third International Workshop on Robot Motion and Control*, pages 319–324, 2002.

[188] D. Kragic, L. Petersson, and H. I. Christensen. Visually Guided Manipulation Tasks. *Robotics and Autonomous Systems*, 40(2-3):193–203, 2002.

[189] G. Kress and H. Almaula. Sensorimotor Requirements for Teleoperation. Technical Report R-62799, FMC Corporation, San Diego, California, 1988.

[190] C. Kube and H. Zhang. Collective Robotic Intelligence. In *From Animals to Animats: International Conference on Simulation of Adaptative Behaviour*, pages 460–468, 1992.

[191] C. Kube and H. Zhang. The Use of Perceptual Cues in Multi-Robot Box-Pushing. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2085–2090, 1996.

[192] R. Kurazume and S. Hirose. Development of a Clining Robot System with Cooperative Positioning System. *Autonomous Robots*, 9(3):237–246, 2000.

[193] Telecom Italia Lab. JADE - Java Agent DEvelopment Framework. http://jade.tilab.com/. Last Visited: 10/09/2007.

[194] R.T. Laird, M.H. Bruch, M.B. West, D.A. Ciccimaro, and H.R. Everett. Issues in Vehicle Teleoperation for Tunnel and Sewer Reconnaissance. In *Proceedings of the International Conference on Robotics and Automation*, 2001.

[195] M.F. Land. The Roles of Eye Movements in Animals. *Vision: The Approach of Biophysics and Neurosciences*, 11 (Biophysics) of Series on Biophysics & Biocybernetics:237–251, 1999.

[196] C. Lane, C. Carignan, and D. Akin. Advanced Operator Interface Design for Complex Space Telerobots. *Autonomous Robots*, 11(1):49–58, 2001.

[197] K. Langley, T.J. Atherton, R.G. Wilson, and M.H.E. Lacombe. Vertical and Horizontal Disparities from Phase. *Image and Vision Computing*, 9(4):296–302, 1991.

[198] S.M. LaValle, D. Lin, and L.J. Guibas. Finding an Unpredictable Target in a Workspace with Obstacles. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 737–742, 1997.

[199] J. Lawton, B. Young, and R. Beard. A Decentralized Approach to Elementary Formation Manuevers. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2000.

[200] D.N. Lee. A Theory of Visual Control of Braking Based on Information about Time-to-Collision. *Perception*, 5:437—-459, 1976.

[201] D.N. Lee. Visuo-Motor Coordination in Space–Time. In *Advanced Tutorials in Motor Behavior*, pages 281—-293. North-Holland Publishing Co., 1980.

[202] M. Lemay, F. Michaud, D. Letourneau, and J.M. Valin. Autonomous Initialization of Robot Formations. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3018–3023, 2004.

[203] M.A. Lewis and K.H. Tan. High Precision Formation Control of Mobile Robots using Virtual Structures. *Autonomous Robots*, 4:387–403, 1997.

[204] S. Li, J. Boskovic, and R. Mehra. Globally Stable Automatic Formation Flight Control in Two Dimensions. In *Proceedings of the AIAA Guidance, Navigation and Control Conference*, 2001.

[205] Y. Li and X. Chen. Control and Stability Analysis on Multiple Robots Formation. In *Proceedings of the 2nd International Conference on Autonomous Robots and Agents*, pages 158–163, 2004.

[206] S. Liang. *Java(TM) Native Interface: Programmer's Guide and Specification*. Addison-Wesley, 1999.

[207] G. Lidoris and M. Buss. A Multi-agent System Architecture for Modular Robotic Mobility Aids. In *Proceedings of the European Robotics Symposium 2006*, pages 15–26, 2006.

[208] I. Lin, R. Dillmann, and F. Wallner. An Advanced Telerobotic Control System for a Mobile Robot with Multisensor Feedback. *Intelligent Autonomous System*, 4, 1995.

[209] S. Lin and A.A. Goldenberg. Neural-Network Control of Mobile Manipulators. *IEEE Transactions on Neural Networks*, 12(5):1121–1133, 2001.

[210] A. Liptay, J.L. Barron, T. Jewett, and I. Van Wesenbeeck. Optic Flow as an Ultra-Sensitive Technique for Measuring Seedling Growth in Long Image Sequences. *Journal of the American Society for Horticultural Science*, 120(3):379–385, 1995.

[211] R. Liscano, R.E. Fayek, A. Manz, E.R. Stuck, and J. Tigli. Using a Blackboard to Integrate Multiple Activities and Achieve Strategic Reasoning for Mobile-Robot Navigation. *IEEE Expert*, 10(2):24–36, 1995.

[212] H. Liu, T.H. Hong, M. Herman, T. Camus, and R. Chellappa. Accuracy vs Efficiency Trade-offs in Optical Flow Algorithms. *Computer Vision and Image Understanding*, 72(3):271–286, 1998.

[213] H. Liu, T.H. Hong, M. Herman, and R. Chellappa. A Generalized Motion Model for Estimating Optical Flow Using 3-D Hermite Polynomials. In *Proceedings of the IEEE International Conference on Pattern Recognition*, pages 360–366, 1994.

[214] S. Liu, D. Tan, and G. Liu. Distributed Formation Control of Robots with Directive Visual Measurement. In *Proceedings of the IEEE International Conference on Mechatronics & Automation*, pages 1760–1765, 2005.

[215] T. Lozano-Pérez, J.L. Jones, E. Mazer, and P.A. O'Donnell. *HANDEY: A Robot Task Planner*. Series in Artificial Intelligence. The MIT Press, 1992.

[216] T. Lozano-Pérez, J.L. Jones, E. Mazer, P.A. O'Donnell, W.E.L. Grimson, P. Tournassoud, and A. Lanusse. Handey: A Robot System that Recognizes, Plans, and Manipulates. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 843–849, 1987.

[217] B. Lucas and T. Kanade. An Iterative Image Regitration Technique with Applications in Stereo Vision. In *Proceedings of the Proceedures of the DARPA Image Understanding Workshop*, pages 121–130, 1981.

[218] R. Lumia. Using Nasrem for Real-Time Sensory Interactive Robot Control. *Robotica*, 12(2):127–135, 1994.

[219] R. Lundh, L. Karlsson, and A. Saffiotti. Dynamic Configuration of a Team of Robots. In *Proceedings of the 16th European Conference on Artificial Intelligence (Workshop on Agents in Dynamic and Real-Time Environments)*, 2004.

[220] R. Lundh, L. Karlsson, and A. Saffiotti. Can Emil Help Pippi? In *Proceedings of the International Conference on Robotics and Automation (Workshop on Cooperative Robotics)*, 2005.

[221] D. MacKenzie, J. Cameron, and R. Arkin. Specification and Execution of Multiagent Missions. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 51–58, 1995.

[222] B. MacLennan. Synthetic Ethology: an Approach to the Study of Communication. In *Proceedings of the 2nd Interdisciplinary Workshop on Synthesis and Simulation of Living Systems*, pages 631–658, 1991.

[223] P. Maes and R.A. Brooks. Learning to Coordinate Behaviours. In *Proceedings of the 8th National Conference on Artificial Intelligence*, 1990.

[224] S. Mahadevan and J. Connell. Automatic Programming of Behavior-Based Robots using Reinforcement Learning. In *Proceedings of the International Conference on Autonomous Agents and Artificial Intelligence*, pages 8–14, 1991.

[225] E. Malis. Visual Servoing Invariant to Changes in Camera Intrinsic Parameters. In *Proceedings of the International Conference in Computer Vision*, pages 704—-709, 2001.

[226] R. Marin, P.J. Sanz, P. Nebot, and R. Esteller. The Internet-Based UJI Tele-Lab: System Architecture. In *Proceedings of the IEEE Conference on Systems, Man & Cybernetics*, pages 4904–4909, 2003.

[227] R. Marin, P.J. Sanz, P. Nebot, and R. Esteller. Multirobot Internet-Based Architecture for Telemanipulation: Experimental Validation. In *Proceedings of the IEEE Conference on Systems, Man & Cybernetics*, pages 3565–3570, 2003.

[228] R. Marin, P.J. Sanz, P. Nebot, R. Esteller, and R. Wirz. Multirobot System Architecture and Performance Issues for the UJI Robotics TeleLab. In *Proceedings of the 1st IFAC Symposium on Telematics Applications in Automation and Robotics*, 2004.

[229] R. Marin, P.J. Sanz, P. Nebot, and R. Wirz. A Multimodal Interface to Control a Robot Arm via Web: A Case Study on Remote Programming. *IEEE Transactions on Industrial Electronics Journal (Special Section on Human-Robot Interface)*, 52(6):1506–1520, 2005.

[230] D. Marr. *Vision*. Freeman Publishers, 1982.

[231] C.S. Marsella, J. Adibi, Y. Al-Onaizan, G.A. Kamink, I. Muslea, and M. Tambe. On Being a Teammate: Experiences Acquired in the Design of RoboCup Teams. *Journal of Autonomous Agents and Multi-Agent Systems*, 4(1–2):440–444, 2001.

[232] M. Mataric. Issues and Approaches in the Design of Collective Autonomous Agents. *Robotics and Autonomous Systems*, 16(2-4):321–331, 1995.

[233] M.J. Mataric. Behavior-Based Control: Main Properties and Implications. In *Proceedings of the Workshop on Intelligent Control Systems, International Conference on Robotics and Automation*, 1992.

[234] M.J. Mataric. Designing Emergent Behaviors: from Local Interactions to Collective Intelligence. In *Proceedings of the 2nd International Conference on Simulation of Adaptive Behavior*, pages 432–441, 1992.

[235] M.J. Mataric. Integration of Representation into Goal-Driven Behaviour-Based Robots. *IEEE Transactions on Robotics and Automation*, 8(3):304–312, 1992.

[236] M.J. Mataric. Minimizing Complexity in Controlling a Mobile Robot Population. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 830–835, 1992.

[237] M.J. Mataric. Interaction and Intelligent Behavior. Technical Report AITR-1495, Massachusetts Institute of Technology, 1994.

[238] M.J. Mataric. Behavior-Based Robotics as a Tool for Synthesis of Artificial Behavior and Analysis of Natural Behavior. *Trends in Cognitive Science*, 2(3):82–87, 1998.

[239] M.J. Mataric. New directions: Robotics: Coordination and Learning in Multirobot Systems. *IEEE Intelligent Systems*, 13(2):6–8, 1998.

[240] M.J. Mataric, M. Nilsson, and K.T. Simsarian. Cooperative Multi-Robot Box-Pushing. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 556–561, 1995.

[241] J. Matijevic. Autonomous Navigation and the Sojourner Microrover. *v*, 280(5362):454–455, 1998.

[242] T. Matsuyama. Cooperative Distributed Vision. In *Proceedings of the 1st International Workshop on Cooperative Distributed Vision*, 1997.

[243] M. Maurer and E.D. Dickmanns. A System Architecture for Autonomous Visual Road Vehicle Guidance. In *Proceedings of the IEEE Conference on Intelligent Transportation System*, pages 578–583, 1997.

[244] C.D. McCarthy. *Performance of Optical Flow Techniques for Mobile Robot Navigation*. PhD thesis, Department of Computer Science and Software Engineering, University of Melbourne,, 2005.

[245] D. McFarland. *Problems of Animal Behaviour*. Longman Scientific and Technical, John Wiley and Sons, 1989.

[246] D. McFarland. Towards Robot Cooperation. In *Proceedings of the 3rd International Conference on Simulation of Adaptive Behavior*, pages 440–444, 1994.

[247] D. McGovern. Experiences and Results in Teleoperation of Land Vehicles. Technical Report SAND 90-0299, Sandia National Laboratories, Albuquerque, New Mexico, 1990.

[248] D. McKenzie, R. Arkin, and J. Cameron. Multiagent Mission Specification and Execution. *Autonomous Robots*, 4(1):29–52, 1997.

[249] B. Mertsching and S. Schmalz. Active Vision Systems. In *Handbook of Computer Vision and Applications. Vol. 3 (Systems and Applications)*, pages 197–219. Academic Press, 1999.

[250] F.G. Meyer. Time-to-Collision from First-Order Models of the Motion Field. *IEEE Transactions on Robotics and Automation*, 10(6):792–798, 1994.

[251] F. Michaud, D. Letourneau, M. Guilbert, and J.M. Valin. Dynamic Robot Formations Using Directional Visual Perception. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2740–2745, 2002.

[252] F. Michaud and M. Vu. Managing Robot Autonomy and Interactivity using Motives and Visual Communications. In *Proceedings of the International Conference on Autonomous Agents*, pages 160–167, 1999.

[253] O. Michel, P. Saucy, and F. Mondada. KhepOnTheWeb: An Experimental Demonstrator in Telerobotics and Virtual Reality. In *Proceedings of the International Conference on Virtual Systems and Multimedia*, 1997.

[254] E. De Micheli, V. Torre, and S. Uras. The Accuracy of the Computation of Optical Flow and of the Recovery of Motion Parameters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(5):434–447, 1993.

[255] Sun Microsystems. Java Remote Method Invocation (Java RMI). http://java.sun.com/javase/technologies/core/basic/rmi/index.jsp. Last Visited: 10/09/2007.

[256] M. Minsky. *The Society of Mind*. Simon and Schuster, 1986.

[257] B. Minten, B. Murphy, and J. Hyams. A Communication-Free Behavior for Docking Mobile Robots. *Distributed Autonomous Robotic Systems*, 4:357–367, 2000.

[258] M. Montemerlo, N. Roy, and S. Thrun. Perspectives on standardization in mobile robot programming: The Carnegie Mellon Navigation (CARMEN) toolkit. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 2436—-2441, 2003.

[259] H.P. Moravec. Toward Automatic Visual Obstacle Avoidance. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, 1977.

[260] H. Mouritsen. Navigation in Birds and other Animals. *Image and Vision Computing*, 19:713–731, 2001.

[261] D.W. Murray and B.F. Buxton. Scene Segmentation from Visual Motion using Global Optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(2):220–228, 1987.

[262] D. Naffin and G. Sukhatme. Negotiated Formations. In *Proceedings of the 8th Conference on Intelligent Autonomous Systems*, pages 181–190, 2004.

[263] P. Nebot and E. Cervera. Agent-based Application Framework for Multiple Mobile Robots Cooperation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1521–1526, 2005.

[264] P. Nebot and E. Cervera. Distributed Optical Flow Navigation using Acromovi Architecture. In *Proceedings of the 15th International Symposium on Measurement & Control in Robotics*, pages XXX–XXX, 2005.

[265] P. Nebot and E. Cervera. A Framework for the Development of Cooperative Robotic Applications. In *Proceedings of the 12th International Conference on Advanced Robotics*, pages 901–906, 2005.

[266] P. Nebot and E. Cervera. Acromovi Architecture: A Framework for the Development of Multirobot Applications. In *Mobile Robots: Moving Intelligence*. Advanced Robotic Systems, 2006.

[267] P. Nebot and E. Cervera. Laboratory on Agent-Based Mobile Manipulation. In P.J. Sanz, R. Marín, and M. Vincze, editors, *Mobile Manipulators: Basic Techniques, New Trends & Applications*, pages DVD–ROM. Publicacions de la Universitat Jaume I, 2006.

[268] P. Nebot and E. Cervera. Optical Flow Navigation over Acromovi Architecture. In *Proceedings of the 3^{rd} International Conference on Informatics in Control, Automation and Robotics*, pages 500–503, 2006.

[269] P. Nebot and E. Cervera. An Integrated Agent-based Software Architecture for Mobile and Manipulator Systems. *Robotica (Special Issue in Mobile Manipulators: basic techniques, new trends and applications)*, 25(2):213–220, 2007.

[270] P. Nebot, D. Gómez, and E. Cervera. TeleLab I: Telelab on Agent-Based Mobile Robots. In R. Marín, P.J. Sanz, and K. Schilling, editors, *Methodology and Tools in Telemanipulation Systems via Internet*, pages DVD–ROM. Publicacions de la Universitat Jaume I, 2004.

[271] S. Negahdaripour and S. Lee. Motion Recovery from Image Sequences using only First Order Optical Flow Information. *International Journal of Computer Vision*, 9(3):163–184, 1992.

[272] R.C. Nelson and J.Y. Alloimonos. Obstacle Avoidance using Flow Field Divergence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11, 1989.

[273] R.C. Nelson and J. Aloimonos. Using Flow Field Divergence for Obstacle Avoidance: towards Qualitative Vision. In *Proceedings of the International Conference on Computer Vision*, pages 188–196, 1988.

[274] P. Nesi, A. Del Bimbo, and D. Ben-Tzvi. A Robust Algorithm for Optical Flow Estimation. *Computer Vision and Image Understanding*, 62(1):59–68, 1995.

[275] W. Newman and M. Lamming. *Interactive System Design*. Addison-Wesley, 1995.

[276] H. Nguyen, N. Pezehkhian, M. Raymond, A. Gupta, and J.M.Spector. Autonomous Communication Relays for Tactical Robots. In *Proceedings of the IEEE International Conference on Advanced Robotics*, pages 35—-40, 2003.

[277] L.A. Nguyen, M. Bualat, L.J. Edwards, L. Flueckiger, C. Neveu, K. Schwehr, M.D. Wagner, and E. Zbinden. Virtual Reality Interfaces for Visualization and Control of Remote Vehicles. *Autonomous Robots*, 11(1):59–68, 2001.

[278] N. Nilsson. Shakey the Robot. Technical Report 323, Artificial Intelligence Center, SRI International, 1984.

[279] F. Noreils and R. Chatila. Plan Execution Monitoring and Control Architecture for Mobile Robots. *IEEE Transactions on Robotics and Automation*, 11(2):255–266, 1995.

[280] A. Oller, E. del Acebo, and J.L. de la Rosa. Architecture for Co-operative Dynamical Physical Systems. In *Proceedings of the 9th European Workshop on Multi-Agent Systems*, 1999.

[281] J. Orwell, S. Massey, P. Remagnino, D. Greenhill, and G.A. Jones. A Multi-Agent Framework for Visual Surveillance. In *Proceedings of the International Conference on Image Analysis and Processing*, 1999.

[282] N. Oswald and P. Levi. Cooperative Vision in a Multi-Agent Architecture. *Lecture Notes In Computer Science*, 1310:709–716, 1997.

[283] L. Overgaard, B.J. Nelson, and P.K. Khosla. A Multi-Agent Framework for Grasping using Visual Servoing and Collision Avoidance. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2456–2461, 1996.

[284] I. Overington. Gradient-Based Flow Segmentation and Location of the Focus of Expansion. In *Proceedings of the Alvey Vision Conferencen*, pages 860–870, 1987.

[285] L. E. Parker. *Heterogeneous Multi-Robot Cooperation*. PhD thesis, M.I.T. Department of Electrical Engineering and Computer Science, 1994.

[286] L.E. Parker. Designing Control Laws for Cooperative Agent Teams. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 582–587, 1993.

[287] L.E. Parker. ALLIANCE: An Architecture for Fault Tolerant Cooperative Control of Heterogeneous Mobile Robots. In *Proceedings of the IEEE/RSJ Intelligent Robots and Systems*, pages 776–783, 1994.

[288] L.E. Parker. L-ALLIANCE: Task-Oriented Multi-Robot Learning in Behaviour-Based Systems. *Journal of Advanced Robotics*, 11(4):305–322, 1997.

[289] L.E. Parker. ALLIANCE: An Architecture for Fault Tolerant Multi-Robot Cooperation. *IEEE Transactions on Robotics and Automation*, 14(2):220–240, 1998.

[290] L.E. Parker. Distributed Autonomous Robotic Systems 4. In *Proceedings of the 5th International Symposium on Distributed Autonomous Robotic Systems*, pages 3–12, 2000.

[291] L.E. Parker. Distributed Algorithms for Multi-Robot Observation of Multiple Moving Targets. *Autonomous Robots*, 12:231–255, 2002.

[292] L.E. Parker. Current Research in Multi-Robot Systems. *Journal of Artificial Life and Robotics*, 7(1-2):1–5, 2003.

[293] J. M. Pasteels, J. Deneubourg, and S. Goss. Self-organization Mechanisms in Ant Societies: Trail Recruitement to Newly Discovered Food Sources. In *From individual to collective behaviour in social insects*. Birkäuser Verlag, 1987.

[294] E. Paulos and J. Canny. Designing Personal Tele-Embodiment. *Autonomous Robots*, 11(1):87–95, 2001.

[295] C. Perkins. *RTP: Audio and Video for the Internet*. Addison-Wesley, 2001.

[296] D. Perzanowski, W. Adams, A. Schultz, and E. Marsh. Towards Seamless Integration in a Multi-Modal Interface. In *Proceedings of the Workshop on Interactive Robotics and Entertainment*, 2000.

[297] L. Petersson, M. Egerstedt, and H.I. Christensen. A hybrid control architecture for mobile manipulation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1285–1291, 1999.

[298] S. Picault and A. Drogoul. The MICRobES Project, an experimental approach towards Open Collective Robotics. In *Proceedings of the 5th International Symposium on Distributed Autonomous Robotic Systems*, 2000.

[299] C. Power and T. Boult. Evaluation of an Omnidirectional Vision Sensor for Teleoperated Target Detection and Identification. In *Proceedings of the International Conference on Robotics and Automation*, 2000.

[300] S. Premvuti and S. Yuta. Consideration on the Cooperation of Multiple Autonomous Mobile Robots. In *Proceedings of the IEEE International Workshop of Intelligent Robots and Systems*, pages 59–63, 1990.

[301] J.L. Prince and E.R. McVeighe. Motion Estimation from Tagged MR Image Sequences. *IEEE Transactions on Medical Images*, 11(2):238–249, 1992.

[302] P. Questa, E. Grossmann, and G. Sandini. Camera Self Orientation and Docking Maneuver using Normal Flow. In *Proceedings of the SPIE AeroSense'95*, pages 274–283, 1995.

[303] W. Rahim. Feedback Limited Control System on a Skid-Steer Vehicle. In *Proceedings of the ANS Fifth Topical Meeting on Robotics and Remote Systems*, 1993.

[304] D. Regan and K.I. Beverley. How do we avoid Confounding the Direction we are Looking and the Direction we are Moving. *Science*, 215:194–196, 1982.

[305] M.I. Rekleitis, G. Dudek, and E.E Milios. Graph-Based Exploration using Multiple Robots. In *Distributed Autonomous Robotics Systems 4*. Springer-Verlag, 2000.

[306] P. Renaud, E. Cervera, and P. Martinet. Towards a Reliable Vision-Based Mobile Robot Formation Control. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3176–3181, 2004.

[307] C Reynolds. Flocks, Herds and Schools: a Distributed Behavioral Model. *Computer Graphics*, 21(4):25–34, 1987.

[308] P. Riley and M. Veloso. On Behavior Classification in Adversarial Environments. *Distributed Autonomous Robotic Systems*, 4:371–380, 2000.

[309] ActivMedia Robotics. ACTS - ActivMedia Color Tracking System. http://robots.mobilerobots.com/ACTS/. Last Visited: 10/09/2007.

[310] ActivMedia Robotics. Advanced Robotics Navigation and Localization (ARNL). http://robots.mobilerobots.com/ARNL/. Last Visited: 10/09/2007.

[311] ActivMedia Robotics. ARIA - Advanced Robot Interface for Applications. http://robots.mobilerobots.com/ARIA/. Last Visited: 10/09/2007.

[312] ActivMedia Robotics. VisLib, a High-Performance Vision Procession Library. http://robots.mobilerobots.com/vislib/. Last Visited: 10/09/2007.

[313] A. Rognone, M. Campani, and A. Verrin. Identifying Multiple Motions from Optical Flow. In *Proceedings of the European Conference on Computer Vision*, pages 258–266, 1992.

[314] S. Rougeaux, L. Berthouze, and Y. Kuniyoshi. Active Calibration of Space Variant Sensors. In *Proceedings of the International Conference on Computational Engineering in Systems Application*, pages 576–580, 1996.

[315] D. Rus, B. Donald, and J. Jennings. Moving Furniture with Teams of Autonomous Robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 235–242, 1995.

[316] D. Rus and M. Vona. A Physical Implementation of the Self-Reconliguring Crystalline Robot. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1726–1733, 2000.

[317] S.J. Russell and P. Norvig. *Artificial Intelligence: a Modern Approach*. Prentice Hall, 2003.

[318] J. Santos-Victor and G. Sandini. Uncalibrated Obstacle Detection using Normal Flow. *Machine Vision and Applications*, 9(3), 1996.

[319] J.A. Santos-Victor and G. Sandini. Visual-Based Obstacle Detection: a Purposive Approach using the Normal Flow. In *Proceedings of the International Conference on Intelligent Autonomous Systems*, 1995.

[320] J.A. Santos-Victor, G. Sandini, F. Curotto, and S. Garibaldi. Divergent Stereo in Autonomous Navigation: from Bees to Robots. *International Journal of Computer Vision*, 14:159–177, 1995.

[321] M. Sarcinelli-Filho, T. Bastos-Filho, and R. Freitas. Mobile Robot Navigation via Reference Recognition based on Ultrasonic Sensing and Monocular Vision. In *Proceedings of the International Conference on Advanced Robotics*, pages 204—-209, 2003.

[322] M. Sarcinelli-Filho, H.J.A Chneebeli, E.M.O. Cladeira, and B.M. Silva. Optical Flow-Based Reactive Navigation of a Mobile Robot. In *Proceedings of the 5th IFAC/EURON Symposium on Intelligent Autonomous Vehicles*, 2004.

[323] M. Sarcinelli-Filho, H.J.A. Schneebeli, and E.M.O. Caldeira. Optical Flow-Based Reactive Navigation of a Mobile Robot. In *Proceedings of the 15th IFAC World Congress on Automatic Control*, 2002.

[324] F.E. Schneider, D. Wildermuth, and H.L. Wolf. Motion Coordination in Formations of Multiple Mobile Robots using a Potential Field Approach. In *Distributed Autonomous Robotic Systems*, pages 305–314. Springer, 2000.

[325] M. Seelinger, J.D. Yoder, E.T. Baumgartner, and S.B. Skaari. High-Precision Visual Control of Mobile Manipulators. *IEEE Transactions on Robotics and Automation*, 18(6):957–965, 2002.

[326] M.A. Sharbafi, C. Lucas, A. Mohammadinejad, and M. Yaghobi. Designing a Football Team of Robots from Beginning to End. *International Journal of Information Technology*, 3(2):101–108, 2006.

[327] T. Sheridan. *Telerobotics, Automation, and Human Supervisory Control*. MIT Press, 1992.

[328] R. Siegwart and P. Saucy. Interacting Mobile Robots on the Web. In *Proceedings of the International Conference on Robotics and Automation*, 1998.

[329] R. Simmons. Where in the World is Xavier, the Robot? *Robotics and Machine Perception (Special Issue on Networked Robotics)*, 5(1):5–9, 1996.

[330] K. Simsarian. *Toward Human-Robot Collaboration*. PhD thesis, Computational Vision and Active Perception Laboratory, Swedish Institute of Computer Science, 2000.

[331] A. Singh and P. Allen. Image-Flow Computation: An Estimation-Theoretic Framework and a Unified Perspective. *Computer Vision, Graphics and Image Processing*, 56:152–177, 1992.

[332] IEEE Computer Society. FIPA- The Foundation of Intelligent Physical Agents. http://www.fipa.org/. Last Visited: 10/09/2007.

[333] K. Souhila and A. Karim. Optical Flow Based Robot Obstacle Avoidance. *International Journal of Advanced Robotic Systems*, 4(1):13–16, 2007.

[334] M.E. Spetsakis. Optical Flow Estimation Using Discontinuity Conforming Filters. *Computer Vision and Image Understanding*, 68(3):276–289, 1997.

[335] H. J. Stains. Carnivores. In *Orders and Families of Recent Mammals of the World*. John Wiley and Sons, 1984.

[336] D. Stilwell and J. Bay. Toward the Development of a Material Transport System using Swarms of Ant-like Robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 766–771, 1993.

[337] H.W. Stone. Mars Pathfinder Microrover: A Low-Cost, Low-Power Spacecraft. In *Proceedings of the 1996 AIAA Forum on Advanced Developments in Space Robotics*, 1996.

[338] P. Stone. *Layered Learning in Multiagent Systems: A Winning Approach to Robotic Soccer*. MIT Press, 2000.

[339] P. Stone and M. Veloso. A Layered Approach to Learning Client Behaviors in the Robocup Soccer Server. *Applied Artificial Intelligence*, 12:165–188, 1998.

[340] P. Stone and M. Veloso. Task Decomposition, Dynamic Role Assignment, and Low-Bandwidth Communication for Real-Time Strategic Teamwork. *Artificial Intelligence*, 110(2):241–273, 1999.

[341] P. Stone and M. Veloso. Multiagent Systems: A Survey from a Machine Learning Perspective. *Autonomous Robotics*, 88(3):345–383, 2000.

[342] P. Stone and M. Veloso. A Survey of Multiagent and Multirobot Systems. In *Robot Teams: From Diversity to Polymorphism*, pages 37–92. A K Peters, Ltd., 2002.

[343] M. Subbarao. Bounds on Time-to-Collision and Rotational Component from First-Order Derivatives of Image Flow. *CVGIP: Image Understanding*, 50:329–341, 1990.

[344] G.S. Sukhatme and J.F. Montgomery. Experiments with Cooperative Aerial-Ground Robots. In *Robot Teams: From Diversity to Polimorphism*, pages 345–368. A K Peters, Ltd., 2002.

[345] V. Sundareswaran. A Fast Method to Estimate Sensor Translation. In *Proceedings of the European Conference on Computer Vision*, pages 257–263, 1992.

[346] I. Suzuki and M. Yamashita. Agreement on a Common x-y Coordinate System by a Group of Mobile Robots. In *Proceedings of the Dagstuhl Seminar on Modeling and Planning for Sensor-Based Intelligent Robot Systems*, 1996.

[347] M. Swain and M. Stricker. Promising Directions in Active Vision. *International Journal of Computer Vision*, 11(2):109–126, 1993.

[348] P. Tabuada, G.J. Pappas, and P. Lima. Feasible Formations of Multi-Agent Systems. In *Proceedings of the American Control Conference*, 2001.

[349] K.A. Tahboub. A Semi-Autonomous Reactive Control Architecture. *Journal of Intelligent and Robotic Systems*, 32(4):445—459, 2001.

[350] D.F. Tello, T.F. Bastos-Filho, M. Sarcinelli-Filho, and C.M. Soria R. Carelli. Optical Flow Calculation Using Data Fusion with Decentralized Information Filter. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 2853–2858, 2005.

[351] G. Terrien, T. Fong, C. Thorpe, and C. Baur. Remote Driving with a Multisensor User Interface. In *Proceedings of the 30th International Conference on Environmental Systems*, 2000.

[352] N. Tinbergen. *The Study of Instinct*. Oxford University Press, 1951.

[353] N. Tinbergen. *The Animal in its World; Explorations of an Ethologist, 1932-1972*. Harvard University Press, 1972.

[354] M. Tiscarelli and G. Sandini. Estimation of Depth from Motion using an Anthropomorphic Visual Sensor. *Image and Vision Computing*, 8(4):271–278, 1990.

[355] X. Tu and D. Terzopoulos. Artificial Fishes: Physics, Locomotion, Perception, Behavior. In *Proceedings of the International Conference and Exhibition on Computer Graphics and Interactive Techniques*, pages 43–50, 1994.

[356] C. Unsal and P.K. Khosla. Mechatronic Design of a Modular Self-Reconfiguring Robotic System. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1742–1747, 2000.

[357] H. Utz, S. Sablatnög, S. Enderle, and G.K. Kraetzschmar. Miro – Middleware for Mobile Robot Applications. *IEEE Transactions on Robotics and Automation (Special Issue on Object-Oriented Distributed Control Architectures)*, 18(4):493–497, 2002.

[358] S.J. Vestli and N. Tschiehold-Gurman. MOPS, a System for Mail Distribution in Office Type Buildings. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 489–496, 1996.

[359] R. Vidal, O. Shakernia, H. Jin, D. Hyunchul, and S. Sastry. Probabilistic Pursuit-Evasion Games: Theory, Implementation and Experimental Evaluation. *IEEE Transactions on Robotics and Automation*, 18(5):662–669, 2002.

[360] B.J.W. Waarsing, M. Nuttin, and H. Van Brussel. Behaviour-based mobile manipulation: the opening of a door. In *Proceedings of the 1st International Workshop on Advances in Service Robotics*, pages 168–175, 2003.

[361] P.K.C. Wang. Navigation Strategies for Multiple Autonomous Mobile Robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 486–493, 1989.

[362] P.K.C. Wang. Navigation Strategies for Multiple Autonomous Robots Moving in Formation. *Journal of Robotic Systems*, 8(2):177–195, 1991.

[363] Z. Wang, Y. Kimura, T. Takahashi, and E. Nakano. A Control Method of a Multiple Non-Holonomic Robot System for Cooperative Object Transportation. *Distributed Autonomous Robotic Systems*, 4:447–456, 2000.

[364] D. Wettergreen, H. Pangels, and J. Bares. Behavior-based Gait Execution for the Dante II Walking Robot. In *Proceedings of the IEEE Conference on Intelligent Robots and Systems*, 1995.

[365] B. Wilcox, B. Cooper, and R. Salo. Computer Aided Remote Driving. In *Proceedings of the Association for Unmanned Vehicle Systems Conference*, 1986.

[366] E. O. Wilson. *The Insect Societies: their Origin and Evolution*. Nacourt, Brace & Co., 1971.

[367] E. O. Wilson. *Sociobiology: the New Synthesis*. Harvard University Press, 1975.

[368] M. Wooldridge. *An Introduction to Multiagent Systems*. John Wiley & Sons Ltd., 2002.

[369] H. Yamaguchi. Adaptive Formation Control for Distributed Autonomous Mobile Robot Groups. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2300–2305, 1997.

[370] A. Yamashita, M. Fukuchi, and J. Ota. Motion Planning for Cooperative Transportation of a Large Object by Multiple Mobile Robots in a 3D Environment. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3144–3151, 2000.

[371] M. Yim, D.G. Duff, and K.D. Roufas. Polybot: a Modular Reconfigurable Robot. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 514–520, 2000.

[372] E. Yoshida, T. Arai, J. Ota, and T. Miki. Effect of Grouping in Local Communication System of Multiple Mobile Robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 808–815, 1994.

[373] E. Yoshida, S. Murata, S. Kokaji, K. Tomita, and H. Kurokawa. Micro Self-Reconfigurable Robotic System using Shape Memory Alloy. *Distributed Autonomous Robotic Systems*, 4:145–154, 2000.

[374] H. Zheng and S.D. Blostein. An Error-Weighted Regularization Algorithm for Image Motion-Field Estimation. *IEEE Transactions on Image Processing*, 2(2):246–252, 1993.

[375] Q. Zheng and R. Chellappa. Automatic Feature Point Extraction and Tracking in Image Sequences for Unknown Image Motion. In *Proceedings of the International Conference on Computer Vision*, pages 335–339, 1993.