Tele-robot with Shared Autonomy:

Distributed Navigation Development Framework

Thomas Geerinck*, Eric Colon**, Sid Ahmed Berrabah*, Kenny Cauwerts*

* Department of Electronics and Informatics (ETRO),

Vrije Universiteit Brussel, Brussels, Belgium

** Department of Mechanics

Royal Military Academy, Brussels, Belgium

Corresponding Author: Prof. Hichem Sahli (hsahli@etro.vub.ac.be)

Department of Electronics and Informatics (ETRO), Vrije Universiteit Brussel (VUB)

Pleinlaan 2 - B-1050 Brussel - Belgium

Tel: +32 (2) 629 29 16 Fax: +32 (2) 629 28 83

Abstract

This paper extensively describes the operability of an advanced demonstration platform incorporating reflexive tele-operated control concepts developed on a mobile robot system. By operability, the creation of an opportunity to develop, simulate, tune and test in real-world environment, mobile robot navigation algorithms is meant. During testing in semi-structured environments, the use of an inertial tracker in combination with a head mounted display, improves significantly the situational awareness of the operator, creating a certain feeling of presence at the remote site. Both on hardware as well as on software level, system components have been elaborated to form a modular whole. In order to offer the opportunity to researchers/students for distance development of complex algorithms, emphasis is laid on communication between robot and operator, as well as communication between different system components, resulting in a distributed framework. The name of the framework is CoRoBA, which stands for Controlling Robots with CORBA, a standardized and popular middleware.

1 Introduction

In this paper, it is our intention to extensively describe an advanced demonstration platform, whit the aim to develop, simulate, tune and test in real-world environment, mobile robot navigation algorithms. This work is elaborated on previous work presented in [8]. Both on hardware as well as on software level, system components have been elaborated to form a modular whole. These system functionalities, being tele-robotic applications in real world environment as well as in simulation environment, fit into our developed distributed framework. Many searchers in robotics have been faced with the difficulty of integrating existing systems into new projects and to recycle existing code that has been produced by previous projects. What is consequently needed in robotics are software architectures based on state-of-the-art software engineering techniques like object-oriented languages, software components and software design patterns [5][3] that greatly improve software reusability. Following these perspectives we developed such a framework, named Controlling Robots with CORBA [9] (CoRoBA). The CoRoBA framework has been developed to implement sensor networks and distributed control applications. It is written in C++ and based on CORBA [9], which is a standardized and popular middleware. CoRoBA compares ORCA-1 [3] that also relies on CORBA for its communication engine. While ORCA makes no assumptions about architecture, interfaces and internals, CoRoBA provides a well defined solution for both aspects. CoRoBA relies on well-known Design Patterns [4] and provides a component based development model. In this paper, two robotic applications developed with this framework are described. First, a tele-robotic system allowing an operator to control the robot remotely, incorporating shared autonomy and tele-presence principles. Second, a Java based multi mobile robot simulator (MoRoS3D) that integrates seamlessly in the framework completes the development solution. Simulated robots and sensors implement the same CORBA interfaces as those implemented by real components, allowing to instantaneously switch between simulation and reality. The fundamental idea is achieved being to develop and tune control algorithms in simulation and to replace simulated by real components once satisfying results are reached.

In order to offer the opportunity to researchers/students for distance development of complex algorithms, communication between robot and operator must be explored. Therefore a quite general overview of the concepts of remote control of mobile robots, tele-robotics, is presented.

Tele-robotics, tele-presence, tele-manipulation, tele-operation, tele-service are all terms that describe ways of handling machines and materials remotely. The term tele-operation refers simply to the operation of a vehicle or system over a distance [6]. Traditionally, tele-operation is divided into direct tele-operation and supervisory control [18]. In direct tele-operation the operator closes all the control loops himself. In between autonomous robots and direct tele-operation, levels of supervised or shared autonomy (control) can be created. Depending on the degree the human operator is directly involved to the robot control, some variables or functions are supervised and some are controlled directly by the operator. However, the

need for the operator effort decreases the autonomy of the robot [17] [22] and increases the operator load and the transferred amount of information between the operator and the robot. Poorly designed user interfaces, on the other hand, can result in spatial disorientation, lack of situational awareness, confusion, and frustration [15]. The human-machine interface must be designed so that it maximizes information transfer while minimizing cognitive load [12]. Step-by-step, the tele-operation technology improves towards tele-presence based technology by increasing the sensory feedback, using HMDs, head motion trackers, datagloves, etc. [20]. Tele-presence means that the operator receives sufficient information about the robot and the task environment, displayed in a sufficiently natural way, that the operator feels physically present at the remote site [18]. The optimal degree of immersion required to accomplish a task is still a topic for discussion. Some researchers claim that high-fidelity tele-presence or tele-existence [19] requires feedback using multiple modalities (visual, auditory, haptic) [12]. The most important feedback provider remains the visual system mounted on the robot. A more human viewpoint is created by using stereovision. Vision is the sensor which is able to give the information ”what” and ”where” for the objects a robot is likely to encounter most completely. These approaches work fine when there is little delay in communication, however once transmission delay is introduced the systems become unstable. The instability problem can be eliminated by shared (supervisory) control. Under shared control the operator gives a goal to the tele-robot. The goal is then executed by a local control system and supervised by the operator. This emerging field of Human-Robot Interaction (HRI) represents an interdisciplinary effort that addresses the need to integrate human factors, cognitive science and usability concepts into the design and development of robotic technology. As the physical capabilities of robots improve, the reality of using them in everyday locations such as offices, factories, homes and hospitals, as well as in more technical environments such as space stations, distant planets, ocean floors and battlefields, is quickly becoming more feasible [21]. Generally speaking, robots are more adept at making some decisions by themselves than others [16]. Unstructured decision making, however, remains the domain of humans, especially whenever common sense is required. In order for robots to perform better, therefore, they need to be able to take advantage of human skills (perception, cognition, etc.) and to benefit from human advice and expertise. To do this, robots need to function not as passive tools, but rather as active partners. Numerous robot control architectures have addressed the problem of mixing humans with robots. Systems like adjustable autonomy, mixed initiative systems [11] and collaborative control [6] [7] have recently received considerable research attention. As an example of adjustable autonomy obstacle avoidance behavior [1] can be considered.

The main contributions of this manuscript lie in describing the distributed framework features, and presenting a functionality scheme of the global system, incorporating the mentioned state of the art hardware devices and software algorithms. This way an advanced demonstration platform is created, offering the possibility for distance development, simulating, tuning and testing in real-world environments of mobile robot navigation algorithms. The remainder is organized as follows. In section 2 the development framework, CoRoBA, is discussed. In section 3 an overview of the system architecture is given, explain-

ing all the co-existing component functionalities, both on hardware and software level. Section 4 presents some results and finally section 5 gives some conclusion.

2 Framework of development

2.1 Middleware selection

The first decision when developing distributed applications concerns the choice of the communication library. Some framework developers have opted for the low-level socket library. While this is a good choice with regard to performance, it is a bad one when dealing with portability and maintenance. A communication Middleware lies somewhere between the low level system API and the application. It automates many operations by abstracting low-level function calls. Among all existing Middlewares, CORBA [9] has been selected because of its language and platform independence. As such a Middleware is quite complex and brings some overheads, one could ask the question if it is really usable for implementing robot control applications. The communication performance can be expressed by the following relation:

$$MessageTransmissionTime = latency + length/datatransferrate \quad (1)$$

On one hand, the operations added by CORBA increases the latency (that is independent of the message length). On the other hand, the extra information contained in a CORBA frame is quite constant (a few hundreds of bytes) and has therefore a larger influence for small data packets. We typically have a 20 to 30% overhead in comparison with raw socket communication [4]. However, with increasing computing power and communication bandwidth, the overhead becomes every day less and less significant. Among different CORBA implementations, we have chosen ACE_TAO because it is widely adopted and supported, it implements most of the CORBA specifications (including Notification Service, AMI, and RT-CORBA) and is free open-source software. But we are not limited to this implementation for developing components and other ORB's (Objects Request Broker) that have links to C++ or other languages like Java and Python can be used too.

2.2 CoRoBA

The Controlling Robots with CORBA (CoRoBA) framework has been developed to implement sensor networks and distributed control applications.

The following design principles are behind the development of CoRoBA. One component implements only one *service*. This design choice has the advantage to clearly separate functionality and to facilitate partitioning of *components*. Each service has a dedicated CORBA *interface* that gives the capability to unambiguously identify each component category. This approach combined with the interface inheritance makes it possible to develop generic and specific tools. Another design

principle is the separation between the process logic and the management logic. This has been achieved by defining separate interfaces: A base interface (*Service*) and one interface for each category of component (*Actuators*, *Processors* and *Sensors*) and by decoupling the management from the process data flow by using different threads. In order to be as exhaustive as possible and not to limit the component capabilities, different operation modes have been defined in the Service interface. In the Periodic mode new data is generated at regular intervals. For components working in synchronous mode, data processing happens when new data is received. The principle of the last mode, the trigger mode, is to control the data processing from an external application.

CoRoBA relies on well-known Design Patterns [4] and provides a component based development model. Components are unit of independent deployment. They are black-boxes only known by their access points called interfaces. These interfaces specify how components and their clients interact. CoRoBA defines three kinds of components according to the classical control paradigm, namely Sensors, Processors and Actuators. Processor components are the key-stone of any control architecture and exhibit the largest potential of reuse while sensors and actuators, that are interfaces between the software and the physical world, are more volatile components. Sensors have connections with the physical world and they output data to one *Event Channel*. They will take input from external non-component sources and introduce it into the application flow. Actuators have output connections with the physical world and received data from one Event Channel. An actuator, as this is the component that will actually execute the ”virtual” descriptions of tasks it receives through its interface. This is the component that based on the data-flow of the application will make the platform move or assure feedback to an operator. Both of the previous components only serve as translators. The actual work of the application is done in the process component. Processors get their inputs from one Event Channel; they transform data and send the result to another Event Channel. This component is the one that will actually process the data and try to deduce things from it, generating other data instead of merely translating it onto another medium. Each component (sensor, processor, actuator) is an independent execution unit whose running cycle is remotely managed through a standardized interface.

The use of this framework simplifies the development of applications. Components developed in simple applications can be reused without any modifications in similar or more elaborated ones. Furthermore, because the framework, that implements many well-known Design Patterns, provides a standardized skeleton for each new application, the code the developer has to write is generally limited to a few tens of lines, that is to the code implementing the useful algorithms.

2.3 Communication between components

CoRoBA proposes two communication modes, a classical synchronous call method and an event based communication scheme. In the former solution, interfaces and operations that are remotely invoked by clients are defined using the CORBA Interface Definition Language (IDL). They are then automatically mapped to code by an IDL compiler. There are many situations where the standard CORBA synchronous request/response model is too restrictive. For instance, clients have to poll

the server repeatedly to retrieve the latest information. Likewise, there is no way for the server to efficiently notify groups of interested clients when data change. For these reasons the Object Management Group introduced the Notification Service that provides a flexible model for asynchronous communication among objects. In CoRoBA components communicate through Events that content data structures formatted according to the CORBA Notification specifications. Events data structure are defined in IDL and mapped to C++ structures by the IDL compiler. The Notification service also provides event filtering and configurability according to various quality of service requirements. Clients of the Notification Service can subscribe to specific events of interest by associating filter objects with the proxies through which the clients communicate with event channels. Furthermore, the Notification Service enables each channel, each connection, and each message to be configured to support the desired quality of service with respect to delivery guarantee, event ageing characteristics, and event prioritization.

The main advantage of event based communication is the decoupling between producers or suppliers and consumers. Consumers can receive events from different producers and producers can send different kinds of events. The advantages of this communication method is counterbalanced by the complicated consumer registration (multiple interfaces, bidirectional object reference handshake, ...).

3 System Overview

3.1 Tele-robotics

Figure 1 shows the different components involved in this application. The processors form the skeleton of the application. Actuator and sensor components only serve as translators. Here the complete demonstration platform will be described as it is used for tele-robotics purposes. As discussed in section 2 the framework CoRoBA makes communication possible between the three types of components. By adding functionalities, functions to the basic version of the components and hence combining them an application can be build. In our tele-robotic application an operator controls a joystick and a head motion tracker. The inputs from the operator are registered by two sensor components, displaying the result on a graphical user interface (GUI), an actuator component. These control commands are also communicated through the distributed framework to specific processors, able to handle them in an intelligent way. An actuator component, communicating with the stereo head, receives the camera commands and camera movement is executed. The joystick controls are fed into the navigation strategy manager. Based on the level of autonomy and the local map of the robot's surroundings, these control commands are adapted. A motion control process communicates these commands to an actuator component interfacing with the robot hardware. The map is generated based on a data fusion process. On its turn this process receives input from a sensor component interfacing with the available sensor setup on the robot. An odometry sensor component keeps track of the robot's motion and updates the robot's position. This information is of importance for the map building and the motion control.

Figure 2 shows the different soft- and hardware modules involved and the exchanged information. It illustrates a dataflow

throughout the system components, demonstrating how abstract sensor readings are transformed into intelligent movement. Both *SensorController* and *MotorController* stand for the programs running on the microcontrollers steering respectively the sensors and motors.

A sensor component retrieves stereo images from the cameras mounted on top of the robot. Processors, compression and decompression, aim at the fast transfer of the images over the distributed framework with respect to their quality. Visual feedback to the operator is provided by means of an actuator component, communicating with a head mounted display (HMD).

In the following all these components will be described at their functionality level. Only the navigation strategy processor component will be conferred more in detail.

The basic mode of operation for the system is traditional or direct tele-operation, including the creation of feeling of presence. In order to introduce shared or supervisory autonomy control aspects to the existing architecture of direct tele-operation, a choice must be made in how to define the responsibilities for both robot and tele-operator. We chose to provide fixed static responsibilities for human and robot. Based on the statement that the aim of robotics is to serve and help humans, our implemented system is well-suited for exploring purposes. The fixed responsibilities are defined in 4 levels of autonomy:

Tele-operation Mode The user has full, continuous control of the robot at low level. The robot takes no initiative except perhaps to stop once it recognizes that communications have failed. It does indicate the detection of obstacles in its path to the user, but will not prevent collision. This is the default autonomy level.

Safe Mode The user directs the movements of the robot, but the robot takes initiative and has the authority to protect itself. For example, it will stop before it collides with an obstacle, which it detects via multiple US and IR sensors.

Shared Control Mode The robot takes the initiative to choose its own path in response to general direction and speed input from the operator. Although the robot handles the low level navigation and obstacle avoidance, the user supplies intermittent input to guide the robot in general directions.

Full Autonomy Mode The robot performs global path planning to select its own routes, acquiring no operator input. The goal of the robot can be specified by the operator or by the robot's vision system by introducing target recognition techniques and tracking.

Note that, the change in autonomy level is made dynamically; whenever the operator desires to change the level of autonomy the robot changes its behavior.

3.1.1 Client Sensors

The main task of this module is the regular update of input commands conferred by the operator. By means of two hardware devices the operator controls the robot and the stereo head. The robot is controlled by use of a joystick, which is interfaced using Direct Input. The stereo head is controlled by movement of the operator's head. A motion tracking device is placed on the head of the operator and registers the rotations of the head made by the operator. The InertiaCube from InterSense is a precision orientation reference system and performs an inertial-based tracking from integration of 9 sensing elements. The range of this device is 360 and has an update rate of 180 Hz. It has 3 DOF (Yaw, Pitch and Roll). However, only 2 of them are actually used, according to the 2 DOF of the pan-tilt stereo head. The range of the pan-tilt stereo head is also limited: 120 of tilt range, 240 of pan range. Whenever the update of input commands is done and the user interface is updated, this new data is sent to the robot.

3.1.2 Client Actuators

The received compressed images from the robot's stereo head, are decompressed. The operator sees a 3D view of the robot's environment, due to the stereovision setup, including the HMD. Being able to look around freely, from a remote location, at a sufficiently high frame rate, due to the image compression, provides the operator with a certain feeling of presence at the remote site.

3.1.3 Robot Sensors & Actuators

The robotic platform is the Nomad200 (Figure 3), an electrical driven mobile robot build by the Nomadic Technologies, Inc. company. Build at early 90's; it has now reached the status of a somewhat antiquated machine in world of robotics. It is equipped with three sensory modules: ultrasonic, infrared and tactile. With its strong and stable structure, the Nomad200 provides an ideal platform for adding extra mechanical and/or sensory structures. On top of the Nomad 200 another PC platform (Figure 3) is placed, linked by a coax cabled ethernet connection.

The robot vision module consists at the hardware level of a stereo head, type Biclops, and two miniature CCD Color Cameras. The head is mounted on the upper PC platform, carrying both cameras (Figure 3). This module performs two tasks concurrently, namely, (1) to accurately control the pan and tilt angle of the stereo head to allow seamless changing the viewpoint, and (2) to capture in a synchronized way frames from the left and right cameras by means of a well suited frame grabber. Further, the captured frames are sent to the remote user. In order to reduce the time needed for the transfer the frames are compressed either using the classical JPEG encoder or a Wavelet based coding technique, compared further on.

In all control modes the operator selects a certain goal or location of interest based on the visual feedback information

received from the robot. The human in the control loop is fully responsible for this goal selection using his own capabilities for active visual search tasks. However, one could think of merging this target choice towards the robot. Therefore biologically inspired visual attention models might be considered for the automated selection of a region of interest, combined with tracking algorithms to keep the target in the field of view. Yet, research must be done to allow useful cooperation between robot visual actions and human visual actions.

3.1.4 Processors

Compression - Decompression

The employed Wavelet based coding scheme, i.e. SQuare Partitioning (SQP) [14] was developed at our department. It allows rate-distortion performances comparable with state-of-the art encoding techniques, allowing lossy-to-lossless reconstruction and resolution scalability. In terms of rate distortion, SQP outperforms JPEG, at considerable compression ratios. Although the encoding with SQP is roughly 4 times slower than with JPEG, still a CIF image can be compressed in real-time at 25 frames per second on a 2Ghz processor.

The compressed frames are communicated via the wireless link to the client. The resolution scalability feature of SQP comes in handy when progressively streaming the data, since the decoder at the client site does not need to wait until all the data has arrived, but it may start reconstructing a lower resolution of the image (from the received data) and start processing that image first while waiting to receive the remaining data that would allow to reconstruct the image at full resolution.

Data Fusion and Map Building

For direct tele-operation the building of a map of the robot's local environment is not indispensable. However, the addition of the mentioned levels of autonomy implicates the need for an accurate representation of the local environment of the robot into a internal obstacle data map. The map is constructed combining the US and IR sensory information. In this context sensor fusion can be defined as the process of combining different sets of, or data derived from, sensory data into a map which represents the environment. Although control architectures with strong reactive characteristics, like the one applied here, do not require an environmental model in order to navigate, the enhanced information using sensor fusion can lead to a more intelligent motion planning.

One of the main motivations for implementing the sensor fusion module is the extension of the spatial coverage. As the number and range of sensors on the robot are limited, not the whole environment of the robot can be scanned at a given moment. The usage of an environmental map enables a memory function, which ensures that the information gathered by past measurements does not get lost. By doing so, successive measurements performed by one and the same sensor can

be used to reduce the uncertainty on the position of an obstacle as the robot moves. However, as not all measurements are reliable, it can be preferable to delete from the map objects that originate from erroneous measurements. For this reason, each object is characterized by the number of spotting and the number of fusion cycles passed since the last spotting, referred to as the ”age” of the object. At the beginning of every fusion cycle, the age of every object on the map is increased by one and then the object is subjected to an elimination test. If the age exceeds a value that depends on the number of spotting, the object is deleted from the map. The dependability itself should be determined experimentally, as it is influenced by the reliability of the sensor measurements as well as the motion speed and cycle time of the sensor fusion module.

Object oriented maps only contain information about the obstacles in the environment. Free space is determined by implication. The main advantage of this kind of mapping is that the whole environment is represented into a small set of data.

In section 3.2 when mentioning fusion of sensor data and map building, we refer further to the method described above.

Navigation Strategy - Motion Controller

According to the selected level of autonomy, the navigation strategy controller selects the proper robot driving behavior. For direct tele-operation, this behavior is straightforward: simply feed the acquired speed and steering commands to the robot's motion controller. In safe mode the available map is checked for collision danger and if necessary an emergency stop is performed. In shared control mode as well as in autonomous mode the robot has the responsibility of the local navigation. To accomplish this task an obstacle avoidance controller is included in the system. The input from the operator can be:

- a finale goal if the way and the time to reach this goal are not very important. In this case the goal point is seen as an attractive point for the robot;
- a finale goal with a set of desired intermediate passing points. The robot tries to reach the target points one by one in the same order they have been introduced by the operator;
- or completely a continuous path until the final goal. In this case, the robot follows the trajectory set up by the user to reach the target point. If an obstacle is detected, the robot uses the obstacle avoidance behavior to bypass it and retrieve its path afterwards.

In all cases the same path planning controller is used and the user has only to define graphically the target point(s). As shown in Figure 4 the output from the obstacle avoidance controller is combined with the input direction from the operator.

The basic building block of the present navigation strategy is a behavior, defined here as a representation of a specific sequence of actions aimed at attaining a given desired objective. Each behavior comprises a set of fuzzy-logic rules. The navigation strategy used in this application is a reactive navigation. It differs from planned navigation in that, while a mission

is assigned or a goal location is known, the robot does not plan its path but rather navigates itself by reacting to its immediate environment in real time. The result of applying iteratively a reactive navigation method is a sequence of motion commands that move the robot from the initial location towards the final location, while avoiding collisions. Example of reactive navigation approaches includes, the Potential Field Methods [10], the Vector Field Histogram [2], and the Nearness Diagram Navigation [13]. In our approach for robot navigation we describe the possible situations by a set of basic rules:

- If no obstacle is detected then use the "Goal seeking behavior". As in figure 7.a which represents two examples of navigation in environments without obstacles with different positions of the target.
- If it's not possible to change direction toward the goal and there is no obstacle in front of the robot, then use the " Go straight ahead behavior" Figure 7.b shows the use of this behavior. A wall bothers the robot in getting toward the target. In this case the robot decides to continue straight ahead until the orientation toward the goal is possible. This behavior avoid the swaying of the robot due to it hesitation.
- If an obstacle is detected in front of the robot and it's still possible to change direction (to turn), then use the " Obstacle Avoidance behavior"
- If there is an obstacle in front of the robot and there is no possibility to change the direction, then use " Make U-turn behavior" as in figure 7.c.

The robot uses a reactive navigation approach by considering the local information from its environment obtained by sonar and infrared sensors. The adaptation of the navigation strategy to the real robot is done through the fuzzy-logic rules parameters (Membership functions, Fuzzification process, and Defuzzification process) of the different behaviors.

A. Goal Seeking Behavior This controller allows the mobile robot, starting from the actual position, to reach a target point. This operation is realized in an environment where there are no obstacles around the robot. Given the azimuth (φ) and the range to the target (ρ), a fuzzy controller calculates the turn angle and speed commands to apply to the robot to reach it. The used controller is of zero order Sugeno's type and uses linguistic decision rules of the form:

If (ρ is Ai) **and** (φ is Bi) **then** ($\Delta\theta$ is Ci)

Where Ai and Bi are fuzzy sets defined respectively in and universes of discourse, and Ci is a constant. In this controller (Figure 5) the change in angle to apply to the robot to reach the target increases as and decreases, i.e. as the target is closer to the robot and far from its direction.

B. Obstacle Avoidance If an obstacle is detected in front of the robot, the nearest point (of this obstacle) to the robot and making the smallest angle (azimuth) with its axis is marked. A fuzzy controller using the information provided by the sensors is initiated. It considers the polar coordinates in the robot frame of the detected points from the obstacles to

estimate the change in angle to apply to the robot to avoid these obstacles. The used controller is a zero order Sugeno's type too and its transfer function is given by Figure 5. In this controller the change in angle to apply to the robot is more important as the obstacle is closer to the robot and closing its way.

C. Go Straight Ahead Behavior This action is used by the robot if there is an obstacle embarrassing it to go toward its goal but no obstacle is detected in front of it. In this case the robot continues moving with its currents speed and orientation.

D. Make U-turn Behavior The robot uses this action in order to leave some blockage situations like a closed way or a narrow way. When this action is activated, the robot makes a U-turn in its position and moves straight ahead until a rotation at the right or at the left is possible.

3.2 Simulation, tuning and testing

Having a simulator has many advantages. First of all it is tremendously cheaper than to buy real robots and sensors. It allows focusing on intelligence and control and disposing of other, less interesting problems. A simulator also increases safety when developing and testing algorithms.

In Figure 6 the concept of integrating our Java based multi mobile robot simulator (MoRoS3D) in the framework is shown. The simulator replaces the real hardware in the control loop in order to test the processor components. These are the keystone of the control architecture and exhibit the largest potential of reuse while sensors and actuators serve as interfaces or translators between the software and other modules. The fundamental idea being to develop and tune control algorithms in simulation and to replace simulated by real components once satisfying results are reached.

The block on the top of Figure 6 named "intelligent control" contains the processors. It is a generic representation that corresponds to the processor components in Figure 1. This part is independent of the platform, real robot or simulator.

The middle block corresponds to the interface components. They make the link between the processors and the physical or simulated world. Sensor and Actuator components implement the same CORBA interfaces as those implemented by real components, allowing to instantaneously switch between simulation and reality.

The last block represents the simulator. It is constituted by different elements that are described hereafter. First of all we need to define models of the physical elements. The robot model deals with the geometric, kinematic and dynamic aspects of the robot. The sensor model encodes information about the sensors like the radiation modes, the minimum and maximum distances, the precision, etc. The environment model contains 3D geometrical representation of the environment. The robot simulator is responsible for the realistic motion of the robot. It receives motion commands from Actuator components. It takes care of the collision with fixed and moving obstacles like other robots. The sensor simulator produces measurement data that are injected in the control loop by the Sensor components.

For its graphical engine, MoRoS3D relies on the Java 3D API. Java 3D offers several ways of defining how an object looks. The most basic way is to work with geometrical shapes and add them together and reshape them to create a complex object. Another is to import a premodeled object from an external file. VRML is one such types supported by Java 3D and allows to draw the objects separately in a drawing program like Wings 3D, which can then export it in VRML format. MoRoS3D allows to place a robot in a 3D environment and to let it interacting with that environment in a manner similar to that of the robot in a real physical situation. Although MoRoS3D visualizes the entire surroundings of the robot, the robot software will only ”see” the information it collects through its sensors, just like with a physical robot. Moreover, motion simulation of multiple robots concurrently, including collision detection, is possible.

Figure 8 represents a simulated autonomous navigation of the Nomad. The MoRoS3D simulator provides different virtual cameras included on-board and tracking ones and provides simple interaction with the user. The user can also specify robots location, reset one or all robots at the same time and erase the path markers. Simple distance sensors, such as Laser, US and IR, are simulated. Sensor simulation is actually a geometric problem that comes to calculating intersections between shapes. This can be easily done with Java3D by using utility classes implementing picking operations. For visual simplicity the sensors are represented by beams instead of cones, although the actual viewing field of some sensors (namely ultrasonic) is cone-shaped.

Besides simulation MoRoS3D can also be used for virtual visualization of a real remote site. In this mode, the pose information of the robot model is updated according to the motion of the real one. Thanks to the event architecture it is straightforward to provide the visualization with the data produced by the Odometry Sensor. The same way, a 3D environment model can be incrementally build based on sensor data produced by the Map Building Processor.

This virtual view of the remote site can serve two purposes.

First, it is very useful for checking what the sensors really see, and if the implemented algorithms are working as they should do. Paths followed by the robots can be visualized and recorded for further analysis and successive trials comparison. Testing and tuning map building and sensor fusion algorithms is a tedious process. Many factors influence the behavior of the robots and running simulations allows to spare time and to consider more combinations. The data from the ultrasonic and the infrared sensors indicate the distance from the sensors to the perceived object. From those data the following information is provided:

- X,Y and G: the coordinates of the position and orientation of the object in a reference frame assigned to the robot. As the position of the sensors is known in the robot’s reference frame, we can compute the position of the spotted objects from the estimated distance.
- ΔX, ΔY and ΔG: Every measurement has a certain range which is bound to contain the actual position of that object.

ΔX, ΔY and ΔG are hence the upper limits of the error on the measurement on the X, Y and G parameter.

- Age: every object is given a certain age, which is actually the number of cycles of the fusion algorithm performed since the object was last detected.
- Spotting: The number of spotting of the object by any of the sensors is also kept, as this value will represent the confidence in the actual presence of the object.

Note that if a sensor has detected no object, the distance of the variable of the type ”Objects” is set to 10000mm. Objects at such distance will not influence the path planning. At this time, the controller only takes into account distances to the three closest obstacles.

Secondly, when using a real robot, the virtual view of the remote site can help the operator to take decisions about future actions. However, this might be a more prospective remark, and well suited for future developments.

Concerning the simulator performances, typical figures for 10 robots with 16 laser distance sensor is 80% processor activity (Centrino 735) and a memory usage of 40MB, image refresh period in this configuration is 80 ms. The executable is stable and does not have memory leaks. Java3D is certainly not the most optimized 3D engine but it is not too slow because all 3D operations actually relies on the 3D rendering library (DirectX or OpenGL), only collision detection and motion control algorithms are executed by the Java engine.

4 Discussion

4.1 Tele-operation performance

The performance of control and visual feedback loops are essential when the performance of the tele-operation system is evaluated. The tele-operation system in the test platform is coordinated, i.e. the position of steering and throttle is transferred to the position of those actuators. When the actuators are speed controlled servos the position control loops are made in the robot's main computer.

Performance of the test vehicle was examined by measuring the step responses of the steering and throttle over the tele-operation loop, ignoring the time delay in this ideal communication test case. The steering and throttle step responses are given in Figure 9. It can be seen that the steering has a relatively big delay, about 2 s to turn 30 degrees. Also the throttle's delay is quite big before reaching the desired speed, 900 ms to attain a speed of 0.5 m/s which is however a quite high speed for a mobile robot. A more acceptable speed of 0.2 m/s is reached after 350 ms. Compared to the mean human reaction time, the speed delay is acceptable. The steering delay however, makes the operator wait for the robot's accurate reaction. To create a reactive system the update frequency of the data input consisting of operator commands is 10 Hz.

The delay in communication is not a constant value. It depends on the distance between router and antenna, and the obstacles in between, e.g. wall of room. However, most of the delay comes from the slowness of the actuators.

The vision loop consists of two processes. First of all the motion-head-tracking-servo-head loop provides some delay (Figure 10: 600ms/50dgr) mainly caused by the servo system. Internally the update rate of the head motion occurs at 180 Hz. The update of these angles to the robot occurs every 100 ms (10 Hz), again to keep this system reactive.

The loop of capturing, compressing and sending stereo images contains two delays. The first delay is the time it takes to compress and decompress, the second is the transmission delay. The time lost with compression and decompression is on the other hand recuperated by the much faster transmission of the images compared to the case without compression. The resolution of the camera is 384 by 288, resulting in a image with size 300Kb, and an obtained frame rate of approximately 5Hz without compression. This is obviously too low to provide good and smooth view of the environment. When using compression a frame rate of approximately 20 Hz is obtained. This augmentation in frame rate is limited by inherent limited processing capacity of the PC.

4.2 Feeling of presence: tele-presence

During real-world testing the operator is provided with a view of the robot's environment. A general problem when applying augmented tele-presence systems are technical complexity and the need for broad bandwidth for data transmission. Use of image compression partly solves this bandwidth issue.

For the simulation of the human active vision system at robot's site, a complex servo system for cameras should have up to 7 DOFs if all head and eye movements were to be tracked and repeated. In practical applications, mostly due to cost and fault probability, the minimum number of DOFs which is still acceptable from the operator point of view is 2.
Ergonomic problems are not minor and need a careful consideration. The problem of simulator sickness (SS) can be significant in tele-presence based tele-operation. The most typical reason is the cue conflict. In cue conflict different nerves get different information from the environment. Typical here is the conflict between visual and vestibular inputs. Other possible reasons can be the resolution of the HMD and the time lags in vision and control resulting in motion blur and a significant decrease of situational awareness. In our system the resolution of the HMD is 800x600 pixels.

Nevertheless, the feeling of presence remains a highly subjective feeling. It is very difficult to design good experiments from which meaningful conclusions can be drawn. It is our believe that a good combination of feedback modalities can provide a sufficient feeling of presence for exploring and surveillance purposes. The HMD provides the operator good quality images, thanks to state of the art compression, at a sufficiently high frame rate, to obtain a smooth view of the environment. Also the stereo vision system provides the operator with a notion of depth, giving him the ability to perceive absolute distances. In order to be constantly aware of the motion of the robot, a layer is placed upon the images in the HMD

with the speed, steer angle, and pan and tilt angles.

We defined a simple line following test, Figure 11, to demonstrate the operability of the platform and highlight possible difficulties. The path of the robot is obviously expressed in the robot's coordinate system. The main issue when designing such a test is the transformation between the robot coordinate system and the real-world coordinate system. This transformation is not known in our simple case implying that the track on the graph is not placed perfectly accurate.

As can be seen on the stereo images below the graph as well as on the graph itself, going straight ahead following a straight line doesn't pose a problem at all. The main difficulty here lies in positioning the robot coinciding the track and consequently drive straight ahead. The delays present in the system don't influence the operability performance in this trivial case.

Difficulties arise when sharp cornering occurs. It is indeed impossible for the human operator to see the feet of the robot in its current configuration. Collisions might happen when cornering to early because of the lack of situational awareness in the robot's direct surrounding. It is clear that experience in driving the robot system, dealing with all the delays including the reaction time delay from the operator itself, is a primordial factor. The solution to overcome this setback is straightforward in our approach of developing a mobile robot platform. Usage of the robot's onboard sensor suit to prevent collisions with obstacles. In similar driving situations, the mentioned levels of autonomy, where the robot receives a certain responsibility, namely safe mode and shared autonomy mode, prove their usability.

5 Conclusion

In this paper, an advanced mobile robot platform is presented. Central aspect in the presentation is the development of a modular distributed component-based framework, CoRoBA, based on CORBA principles. Two robotic applications have been elaborated based on this framework. First, a tele-robotic application can be used for exploration and surveillance purposes. At present a manual control system by the human operator using a joystick has been developed. In order to reduce the operator's workload related to the robot's control, we introduced shared autonomy principles by defining several levels of autonomy. An obstacle avoidance algorithm based on fuzzy logic has been implemented, giving the robot the responsibility of the local navigation. The current stereo vision system allows the human operator not only to feel that he/she is located at the remote task area, but also to catch a sense of distance to the objects existing in the environment. This is accomplished by integrating different technologies, such as image compression, wireless communication, head motion tracking, etc.

Second, a multi mobile robot simulator, MoRoS3D, integrates seamlessly with the existing intelligent software modules, the processors, through the framework structure, by simply adapting sensor and actuator components. In simulation, crucial parameters and processes influencing the robot's behavior, can be tuned and tested. When a switch to real-world testing has to be done, other sensor and actuator components can be plugged, interfacing with the actual hardware components of the robot.

It is clear that in order to benefit from the facilities offered by the framework, the developer has to accept to use at least

CORBA as communication middleware. For the moment this framework is in its consolidation phase and is used internally by the robotics laboratory of the RMA and the VUB.

The fuzzy controllers for robot navigation are tuned based on human knowledge (expert information) which is not precise. Another important portion of information in our system is numerical information, which is collected from sensors (sonar sensors, infrared sensors) or can be obtained according to physical laws. To exploit this information, we propose to use, in the future, an adaptive fuzzy system which is a fuzzy logic system equipped with a training algorithm. The fuzzy logic system is constructed from a collection of fuzzy IF-THEN rules based on the human linguistic knowledge and the training algorithm adjusts the parameters (and the structures) of the fuzzy logic system based on numerical input-output pairs. This adaptive fuzzy system can be viewed as fuzzy logic system whose rules are automatically generated through training.

The fuzzy logic system considered in this case consists of a collection of fuzzy IF-THEN rules with product operations for the rules implication, singleton fuzzifier, center average defuzzifier, and Gaussian membership functions.

Acknowledgments

This research has been conducted within the framework of the Inter-Universitary Attraction-Poles program number IAP 5/06 Advanced Mechatronic Systems, funded by the Belgian Federal Office for Scientific, Technical and Cultural Affairs.

References

[1] J. Borenstein and Y. Koren, *Teleautonomous guidance for mobile robots*, IEEE Trans. on Systems, Man and Cybernetics **20** (1990), no. 6, 1437–1443.

[2] J. Borenstein and Y. Koren, *The vector field histogram – fast obstacle-avoidance for mobile robots*, IEEE Journal of Robotics and Automation **7** (1991), no. 3, 278–288.

[3] A. Brooks, T. Kaupp, A. Makarenko, A. Orebck, and S. Williams, *Towards component-based robotics*, Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2005), 2005, p. In Press.

[4] E. Colon, H. Sahli, and Y. Baudoin, *Coroba, a multi mobile robot control and simulation framework*, Special Issue on "Software Development and Integration in Robotics" of the International Journal on Advanced Robotics (2006), tbp.

[5] S. Enderle, H. Utz, S. Sablatnn, S. Simon, G. Kraetzschmar, and G. Palm, *Miro: Middleware for autonomous mobile robots*, Proceedings of Telematics Application (Weingarten, Germany), July 2001, pp. 149–154.

[6] Terrence Fong and Charles Thorpe, *Vehicle teleoperation interfaces*, Auton. Robots **11** (2001), no. 1, 9–18.

[7] Terrence W Fong, Chuck Thorpe, and C. Baur, *Collaboration, dialogue, and human-robot interaction*, Proceedings of the 10th International Symposium of Robotics Research, Lorne, Victoria, Australia (London), Springer-Verlag, November 2001.

[8] Thomas Geerinck, Valentin Enescu, Ioan Alexandru Salomie, Sid Ahmed Berrabah, Kenny Cauwerts, and Hichem Sahli, *Tele-robots with shared autonomy: tele-presence for high level operability.*, ICINCO, 2005, pp. 243–250.

[9] Michi Henning and Steve Vinoski, *Advanced corba programming with c++*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[10] O. Khatib, *Real-time obstacle avoidance for manipulators and mobile robots*, Int. J. Rob. Res. **5** (1986), no. 1, 90–98.

[11] Julie L. Marble, David J. Bruemmer, Douglas A. Few, and Donald D. Dudenhoeffer, *Evaluation of supervisory vs. peer-peer interaction with human-robot teams.*, HICSS, 2004.

[12] Roger Meier, Terrence W Fong, Chuck Thorpe, and C. Baur, *Sensor fusion based user interface for vehicle teleoperation*, International Conference on Field and Service Robotics (FSR '99), August 1999.

[13] J. Minguez and L. Montano, *Nearness diagram navigation. a new real-time collision avoidance approach*, IEEE/RSJ Int. Conf on Intellingent Robots and Systems (Takamatsu, JP), 2000.

[14] A. Munteanu, J. Cornelis, G. Van der Auwera, and P. Cristea, *Wavelet-based lossless compression scheme with progressive transmission capability*, International Journal of Imaging Systems and Technology, Special Issue on Image and Video Coding **10** (1999), no. 1, 76–85.

[15] Roberto Olivares, Chen Zhou, Julie A. Adams, and Bobby Bodenheimer, *Interface evaluation for mobile robot teleoperation*, Proceedings of the ACM Southeast Conference (ACMSE03) (Savannah, GA) (Mark Burge, ed.), mar 2003, pp. 112–118.

[16] Jean Scholtz, *Theory and evaluation of human robot interactions.*, HICSS, 2003, p. 125.

[17] Jean Scholtz, Brian Antonishek, and Jeff Young, *Operator interventions in autonomous off-road driving: effects of terrain.*, SMC (3), 2004, pp. 2797–2802.

[18] Thomas B. Sheridan, *Telerobotics, automation, and human supervisory control*, MIT Press, Cambridge, MA, USA, 1992.

[19] S. Tachi, *Telecommunication, teleimmersion and tele-existence*, Ohmsha Ltd., 2003.

[20] S. Tachi, H. Arai, and T. Maeda, *Development of anthromorphic tele-existence slave robot*, Proceedings of the International Conference on Advanced Mechatronics (Tokyo), 1989.

[21] H. A. Yanco and J. L. Drury, *A taxonomy for human-robot interaction*, AAAI Fall Symposium on Human-Robot Interaction, AAAI Technical Report FS-02-03, November 2002, pp. 111–119.

[22] H. A. Yanco, J. L. Drury, and J. Scholtz, *Beyond usability evaluation: Analysis of human-robot interaction at a major robotics competition*, Human-Computer Interaction **19** (2004), 117–149.

List of Figures

6 Figures

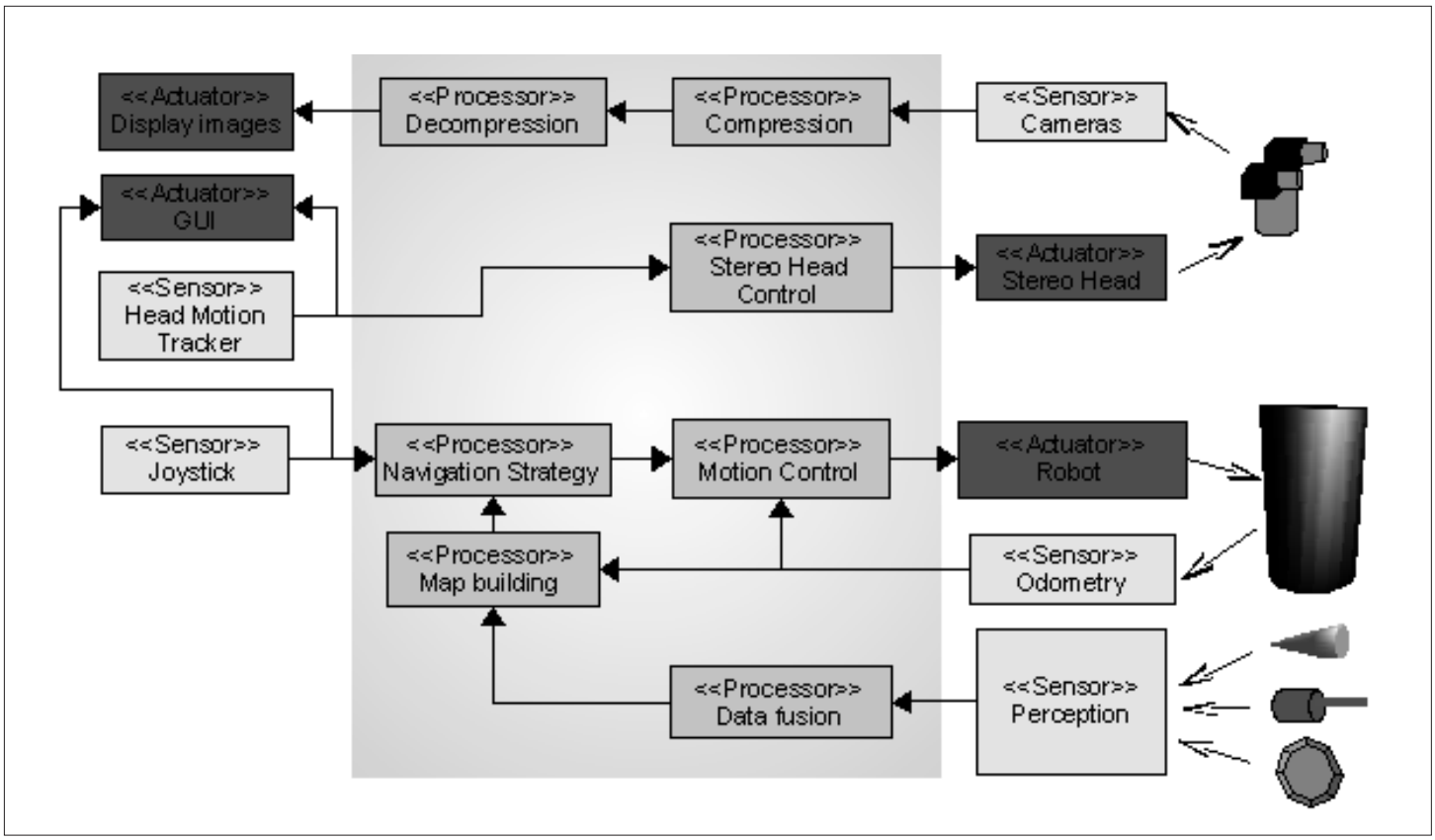


Figure 1. Global system architecture

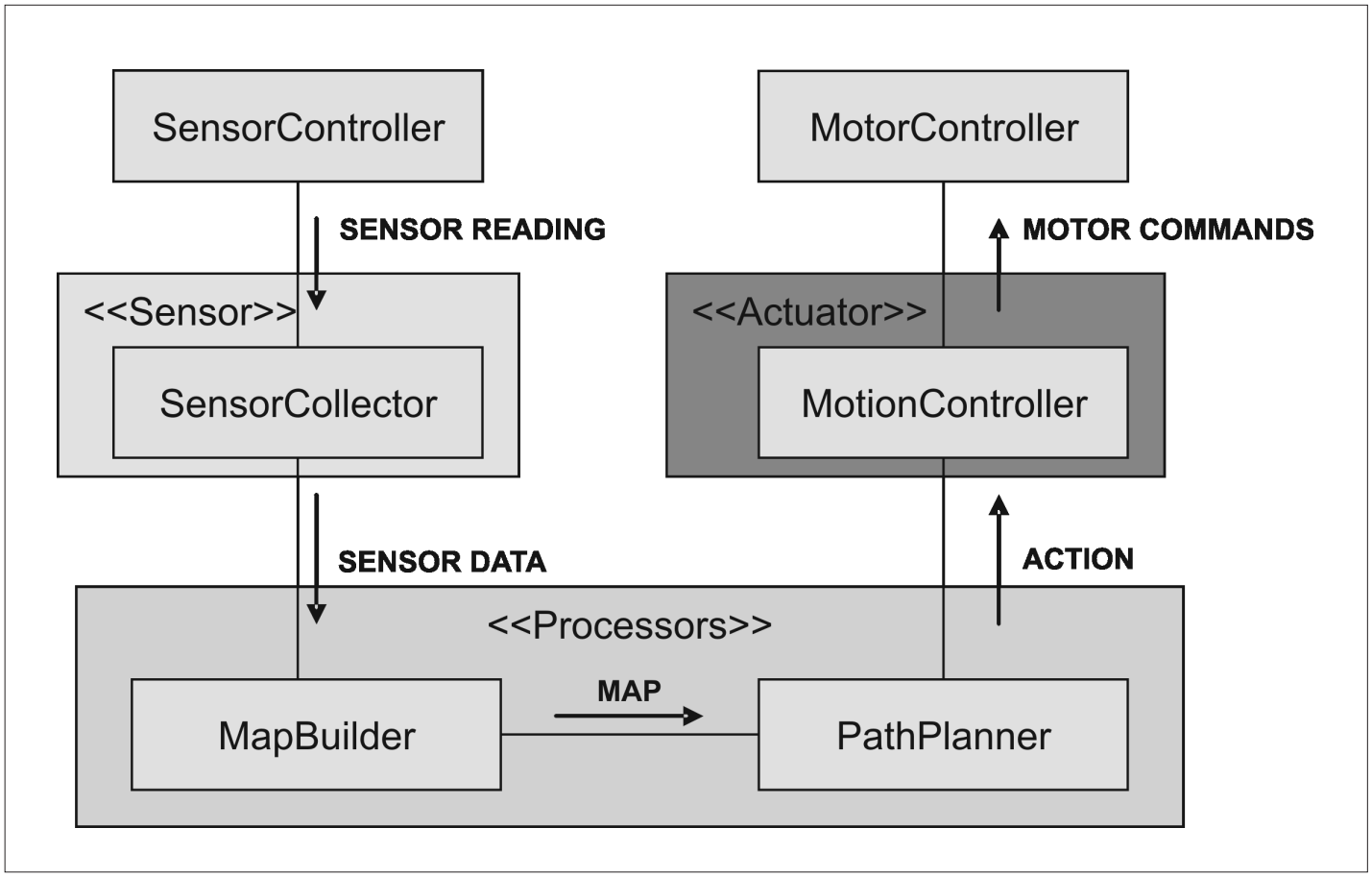


Figure 2. Control software collaboration diagram

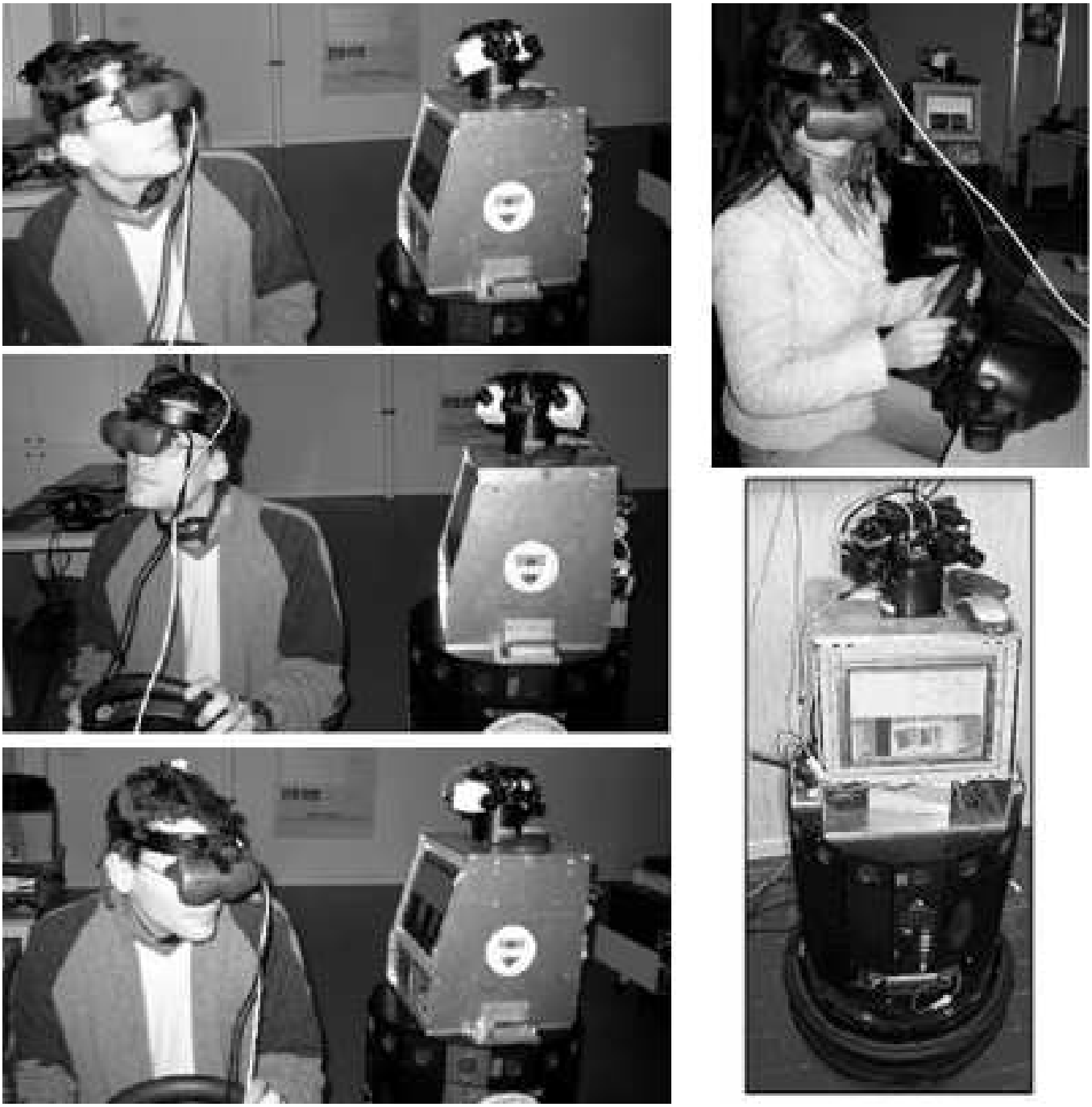


Figure 3. The Nomad200 robot with the upper PC-platform and the biclops head on top. The stereo vision system forms the robot's eyes

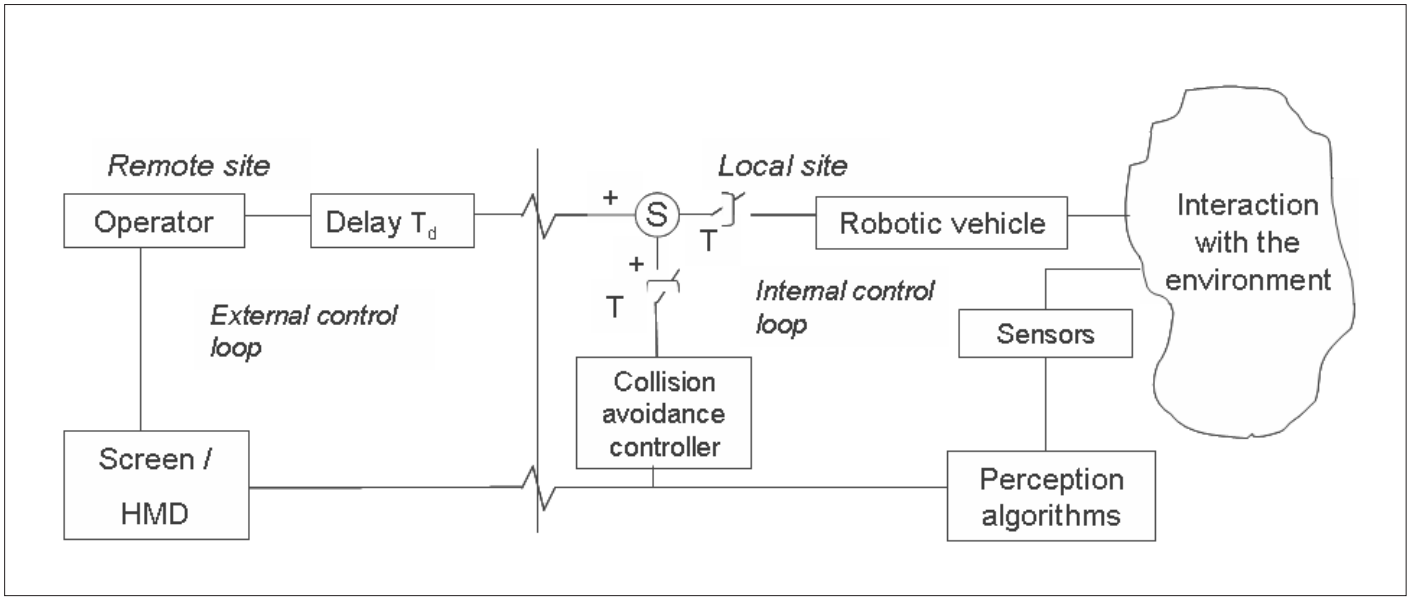


Figure 4. Obstacle avoidance control architecture

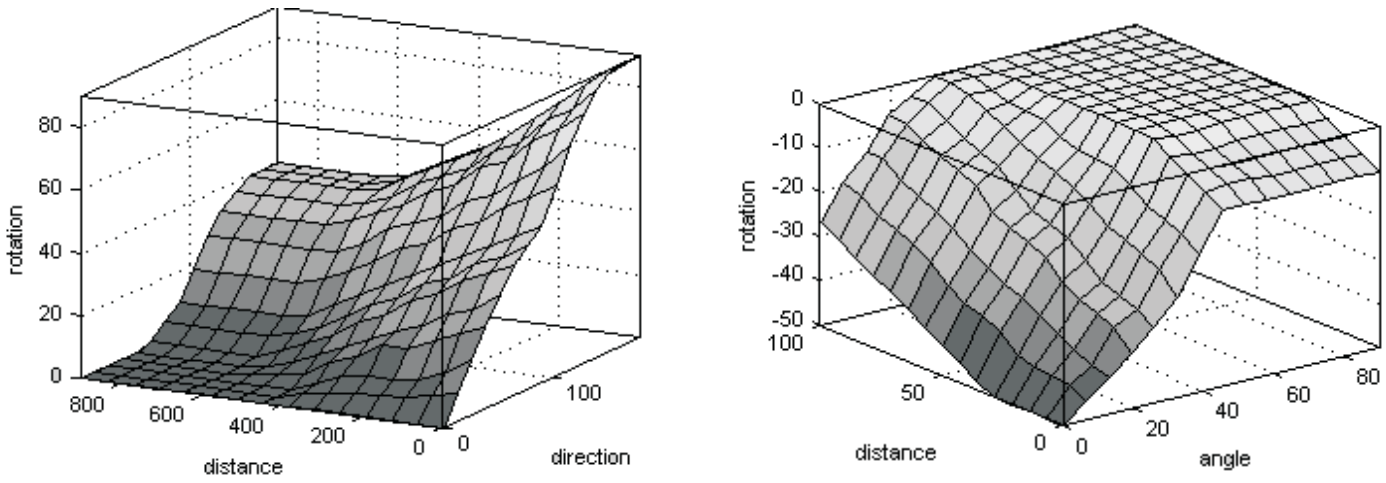


Figure 5. Transfer function of the controller for Goal seeking and Obstacle Avoidance

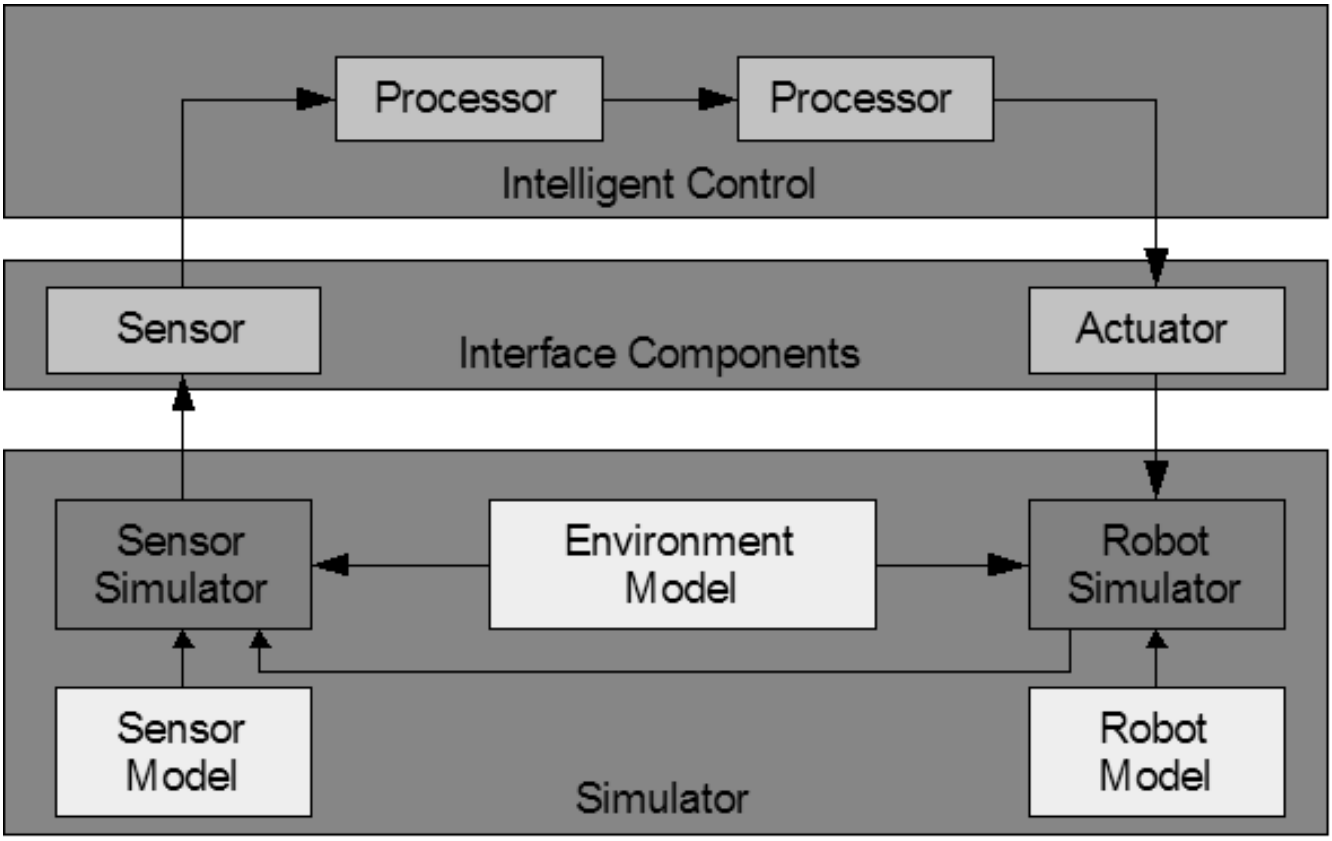
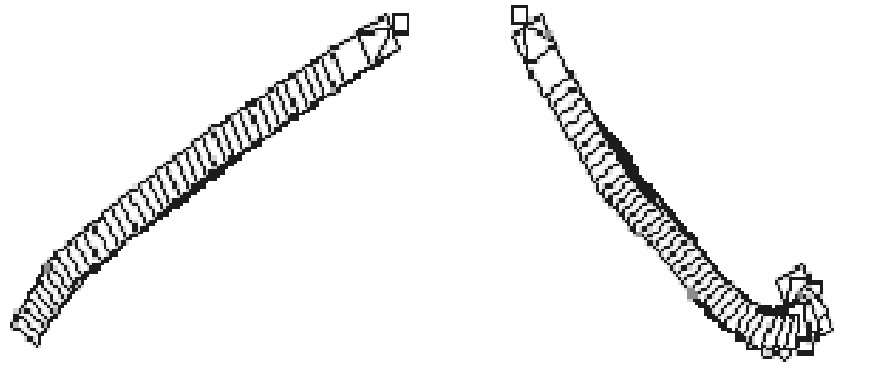
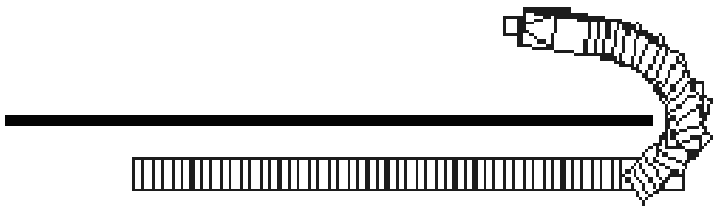


Figure 6. MoRoS3D simulator architecture



a)



b)

c)

Figure 7. path planning

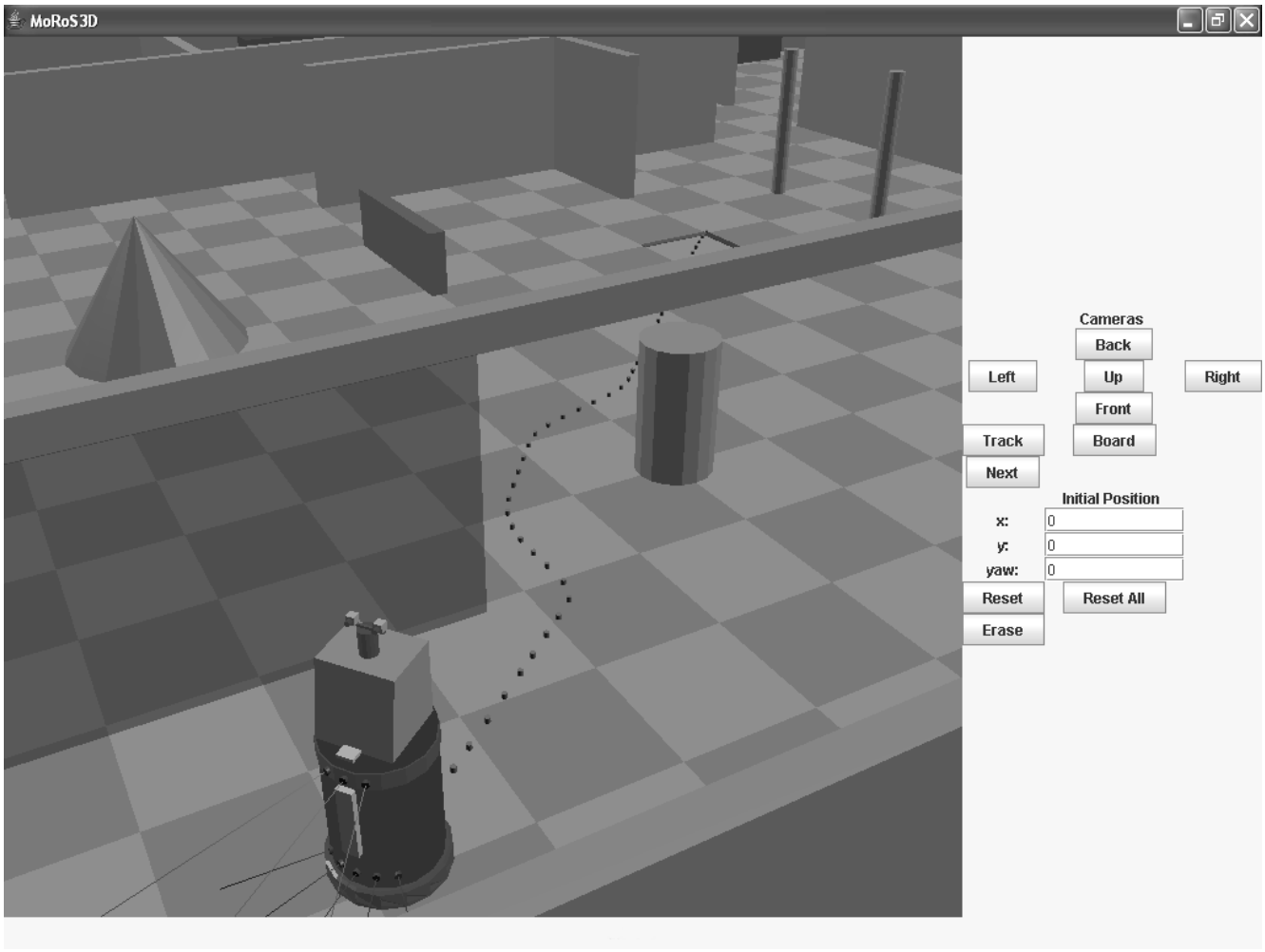


Figure 8. Simulated autonomous navigation in MoRoS3D simulator

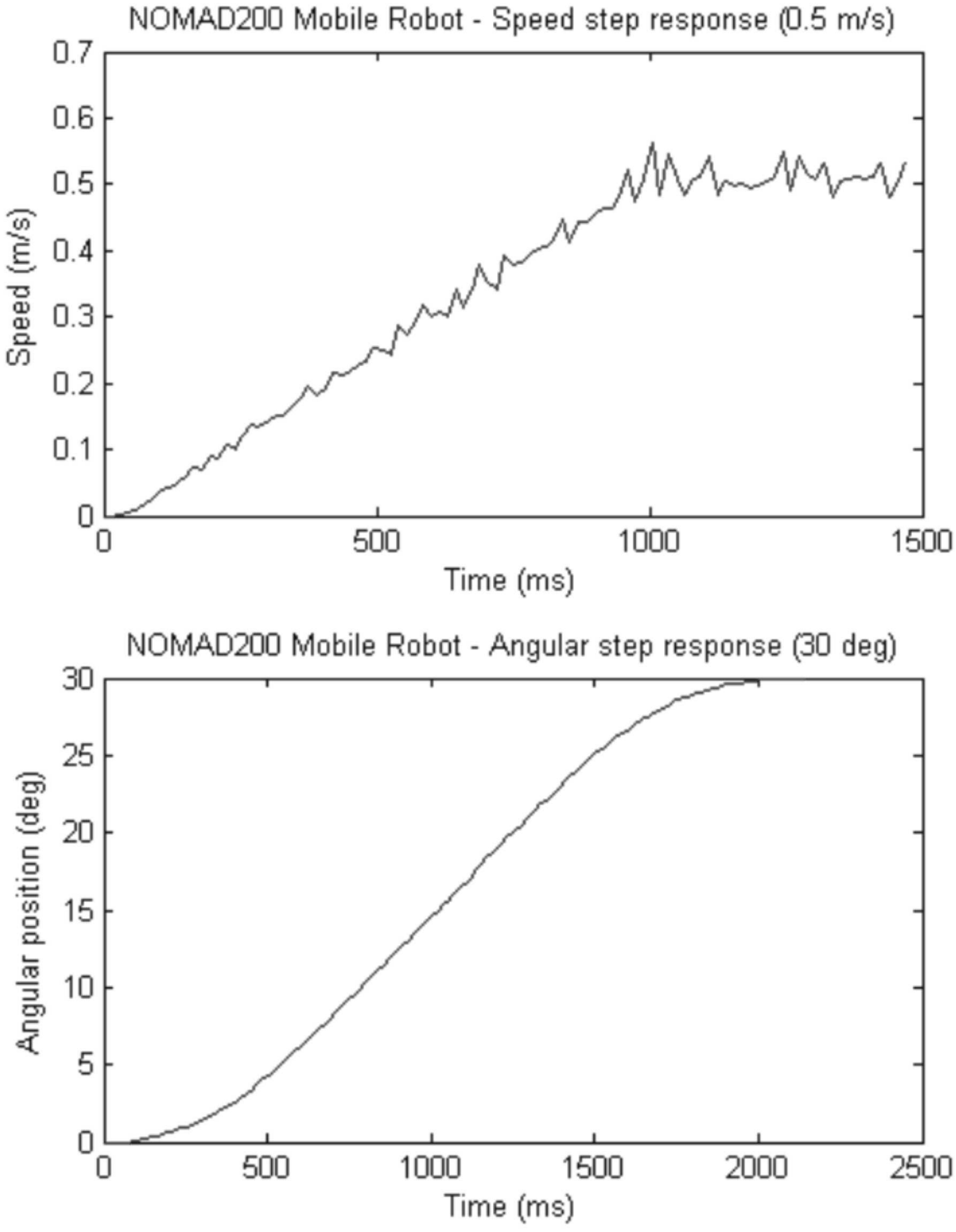


Figure 9. Speed and angular step response from the Nomad200 robot platform

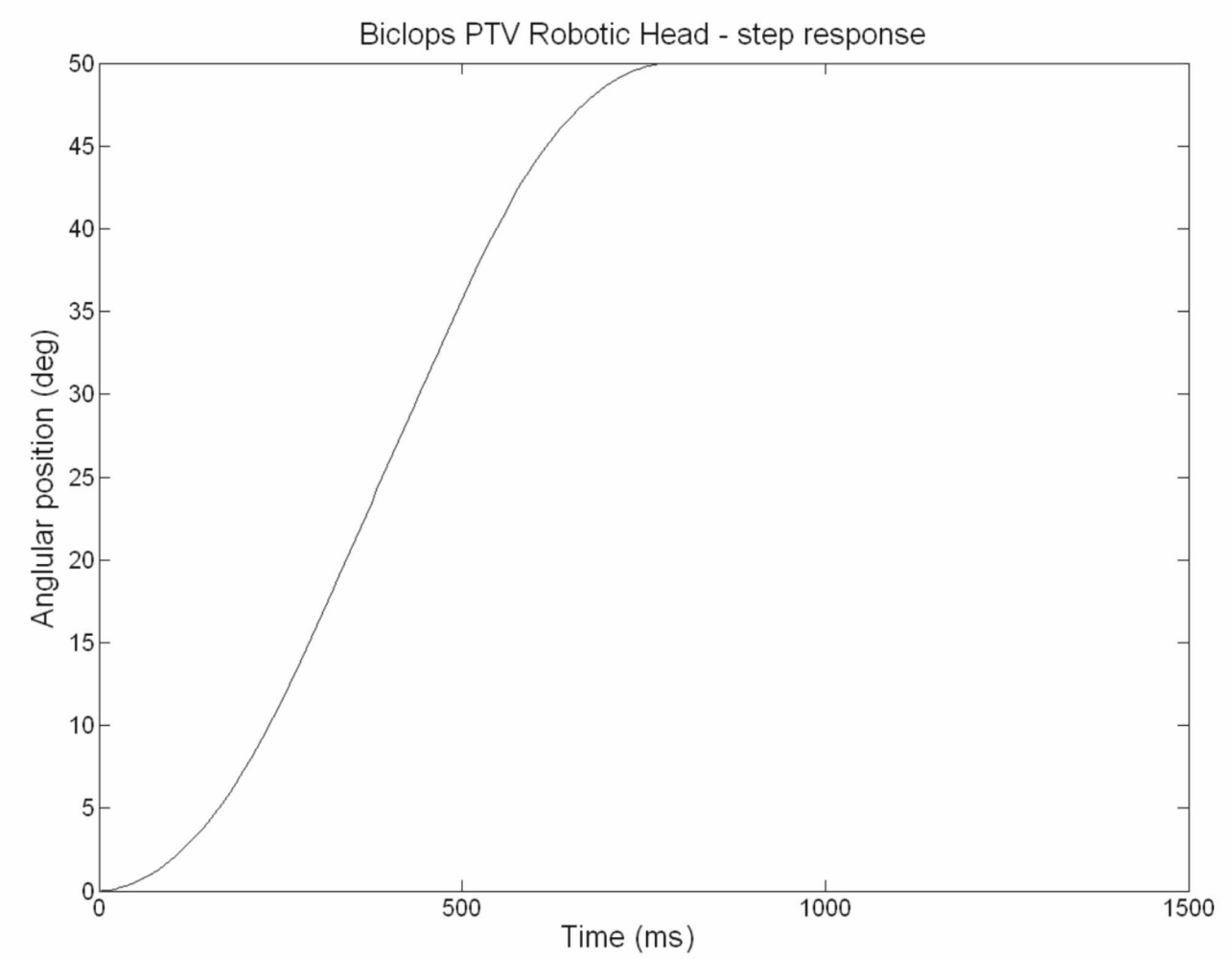


Figure 10. Biclops robotic head step response

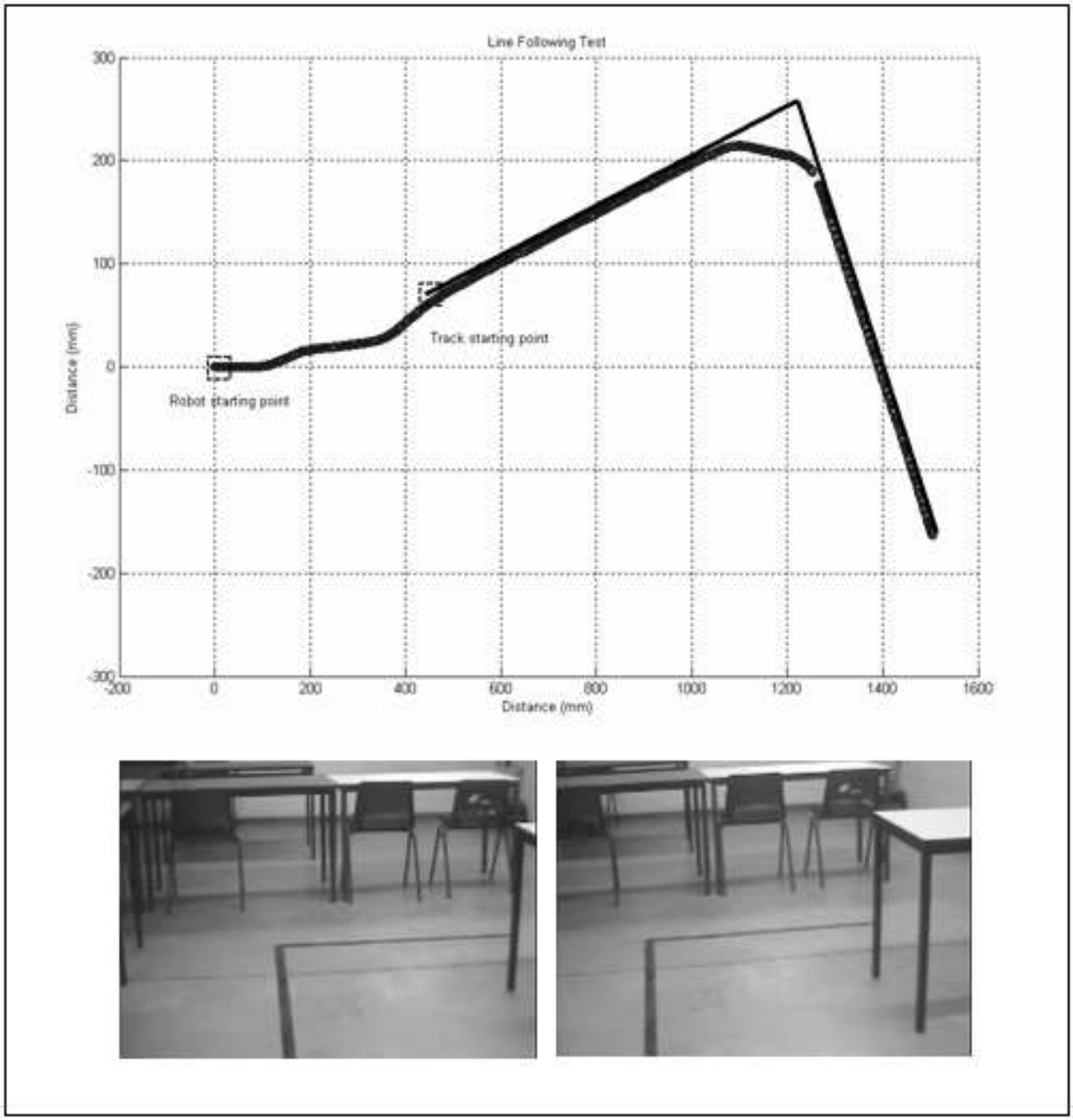


Figure 11. Result of a line following test, with cornering. Below, the stereo images of the simple test course.