Mobile Manipulation: A Case Study

A. HENTOUT¹, B. BOUZOUIA², I. AKLI³ and R. TOUMI⁴

¹, ², ³ Division of Computer-Integrated Manufacturing and Robotics (DPR)
Advanced Technologies Development Centre (CDTA)
BP 17, Baba Hassen, Algiers 16303
Algeria

⁴ Laboratory of Robotics, Parallelism and Electro-energy (LRPE)
University of Sciences and Technology Houari Boumediene (USTHB)
BP 32, El Alia, Bab Ezzouar, Algiers 16111
Algeria

1. Introduction

Classically, manipulators consist of several links connected together by joints. The main purpose in using these robots is to manumit the human from tedious, arduous and repetitive tasks. Nevertheless, the limited dimensions of the links and the morphology of the fixed-base manipulators, create, therefore, limited accessible workspaces.

To support the development and the new application fields of manipulators, the locomotion had to be combined to the manipulation creating, thus, mobile manipulators. This kind of robots consists of coupling manipulation (represented by a manipulator) and locomotion (represented by a mobile base). The conventional structure of this type of robots is a manipulator mounted upon a mobile base. The mobility extends the workspace of the manipulator and increments its operational capability and flexibility (Sugar & Kumar, 1998).

Mobile manipulators allow the most usual missions of robotics that require both abilities of locomotion and manipulation. They have applications in many areas such as grasping and transporting objects, mining, manufacturing, forestry, construction, etc. Recently, target environment for for activity of such robots has been shifting from factory environment to human environment (Nagatani et al., 2002) (offices, hospitals, homes, assistant for disabled and elderly persons, etc.) because they are particularly well suited for human-like tasks (Alfaro et al., 2004).

However, the motion study of these robots is different and more difficult than that of manipulators. Firstly, combining a mobile base and a manipulator creates redundancy. Secondly, the mobile base has a slower dynamic response than the manipulator. Thirdly, the mobile base is often subject to non-holonomic constraints while the manipulator is usually unconstrained. Finally, the task to be carried out by the robot must be decomposed into tiny movements to be executed by the manipulator and large movements to be carried out by the mobile base (Chen et al., 2006).
In recent years, there are a number of researchers studying mobile manipulators control. These studies led to different approaches.

One of the general approaches is to consider the locomotion as extra joints of the manipulator (Nagatani et al., 2002). In this case, the mobile manipulator is regarded as a redundant robot where the redundancy is introduced by the motion of the mobile base (Sasaki et al., 2001). Erden and colleagues (Erden et al., 2004) describe a multi-agent control system to a service mobile manipulator that interacts with human during an object delivery and hand-over task in two dimensions. The identified agents of the system are controlled using fuzzy control. The membership functions of the fuzzy controller are tuned by using genetic algorithms. The authors in (Chen et al., 2006) propose a three-level neural network-based hierarchical controller. The bottom-level controls each joint motor independently. The middle-level consists of a neural network and two sub-controllers. The high-level is a task-planning unit that defines the desired motion trajectories of each degree of freedom (dof).

Colle et al. (Colle et al., 2006) propose a multi-agent system for controlling their mobile manipulator ARPH. For each articulation is affected a reactive agent that realize in parallel a local task without a priori knowledge on the actions of the other agents. Each agent computes the current position of the end-effector and attempts by tiny local movements to match that position with the desired one.

The other type of approaches controls separately the mobile base and the manipulator neglecting the dynamic interaction between the two sub-systems. Such strategies are appropriate when the coupled dynamics is not significant (ex. when the robot moves at low speed) (Chen et al., 2006). The authors in (Waarsing et al., 2003) implement a behaviour-based controller over a mobile manipulator to make it able to open a door. The locomotion control system, the manipulator control system and the sensor systems cooperate in order to realize such a behaviour. Petersson et al. (Petersson et al., 1999) propose an architecture that enables the integration of the manipulator into a behaviour-based control structure of the mobile base. This architecture combines existing techniques for navigation and mobility with a flexible control system for the manipulator.

The robot, as human, must have the ability to obtain information about its environment in order to achieve each step of the manipulation task. The most important sensor which provides rich and varied information on the environment is the vision sensor (the camera) (Trabelsi et al., 2005). Based on hand-eye relation, visual servo system has two types of camera configuration (i) Eye-in-hand configuration and (ii) Eye-to-hand configuration (Flandin et al., 2000). The manipulator behaves as a hand and the camera as its eye. The camera is said as Eye-in-hand when rigidly mounted on the end-effector. Here, there exists a known, often constant relationship between the position of the camera and that of the end-effector. The camera is said as Eye-to-hand when it observes both of the robot and the (Muis & Ohnishi, 2005). Vision-based servoing schemes are flexible and effective methods to control robot motion from camera observations (Hutchinson et al., 1996). Many applications in vision-based robotics, such as mobile robot localization (Blaer & Allen, 2002), object grasping (Muis & Ohnishi, 2005) (Janabi-Sharifi & Wilson, 1998) and manipulation (Trabelsi et al., 2005), handling and transporting objects from one place to another (Trabelsi et al., 2005), navigation (Winter et al., 2000), etc.

This chapter highlights several issues around mobile manipulation in indoor environments. The first aspect consists of planning a coordinated trajectory for the non-holonomic mobile base and the manipulator so that the end-effector of the robot can be as near as possible,
from a predefined operational trajectory. The second aspect deals with a position-based servoing control of mobile manipulators by using an eye-in-hand camera and a LMS sensor. These applications are developed within the framework of control architecture of such robots while taking into account the constraints and difficulties mentioned above. The architecture consists of a multi-agent system where each agent models a principal function and manages a different sub-system of the robot. The unified models of the mobile manipulator are derived from the sub-models describing the manipulator and the mobile base. These applications are considered in the case of the RobuTER/ULM mobile manipulator of the Division of Computer-Integrated Manufacturing and Robotics of the Advanced Technologies Development Centre.

The second section of the chapter describes the hardware and the software architecture of the experimental robotic system. Section three explains the multi-agent architecture proposed to control mobile manipulators. Section four describes the implementation of the control architecture. The agents are implemented as a set of concurrent threads communicating by TCP/IP sockets. In addition, the threads of each agent communicate by shared variables protected by semaphores. The autonomy of decision-making and the cooperation between the agents are presented in section five through two problems. The first one focuses on trajectory planning and control for mobile manipulators. The end-effector of the robot has to follow a predefined operational trajectory (given by a set of Cartesian coordinates \((x, y, z)\)) while the mobile base avoids obstacles present in the environment. The second part of the experiments, in order to give the robot the ability to manipulate in an indoor environment, deals with position-based servoing control of mobile manipulators using a single camera mounted at its end-effector (eye-in-hand camera) and a LMS sensor. Conclusions and future works are presented at the end of the chapter.

2. Architecture of the experimental

The experimental robotic system, given by Fig. 1, consists of a Local (Operator) site and a Remote site, connected by wireless communication systems:

- **Local site**: it includes an off-board PC running under Windows XP, a wireless TCP/IP communication media, a wireless video reception system and input devices.
- **Remote site**: it includes the RobuTER/ULM mobile manipulator, a wireless TCP/IP communication media and a wireless video transmission system.

2.1 Architecture of the RobuTER/ULM mobile manipulator

RobuTER/ULM is composed of a rectangular differentially-driven mobile base on which is mounted a manipulator. The robot is controlled by an on-board MMX industrial PC and by four MPC555 microcontroller cards communicating via a CAN bus. The on-board PC is running under Linux 6.2 with RTAI layer 1.3. This layer interfaces C/C++ application with that developed under SynDEx (http://www.syndex.org). The first MPC555 card controls the mobile base. The second and the third control the first three and the last three joints of the manipulator. The last MPC555 controls the effort sensor.

The mobile base has two driven wheels ensuring its mobility and two free wheels to maintain its stability. The mobile base is equipped with a belt of 24 ultrasonic sensors, a laser measurement system at the front and an odometer sensor on each driven wheel.
The manipulator is a six-dof ultra-light manipulator (ULM) with two-finger electrical gripper. All of the joints are rotatable. The manipulator is equipped with incremental position sensor for each articulation and with a six-dof effort sensor integrated on the gripper.

The robot is also equipped with a monochrome CCD camera placed on the gripper (eye-in-hand camera) with an acquisition card. The resolution of the camera is 352*240 pixels. Images are directly transmitted to the off-board PC via the wireless video transmission system. The camera is maneuverable enough to explore the environment of the robot due to the six dof of the manipulator.

2.2 Kinematic analysis of RobuTER/ULM

2.2.1 Main reference frames

The kinematic analysis of the robot needs to focus on the following main reference frames and transformation matrices (Fig. 2) (Hentout et al., 2009a):
Manipulator
Remote site
Operator site
arm
Wireless video
Camera
sensors
Efforts
sensor
antenna
Omnidirectional
Mission

Fig. 2. Reference frames of the RobuTER/ULM and the transformation matrices

- $R_A = (O_A,\overrightarrow{x_A},\overrightarrow{y_A},\overrightarrow{z_A})$: The absolute reference frame.
- $R_B = (O_B,\overrightarrow{x_B},\overrightarrow{y_B},\overrightarrow{z_B})$: The mobile base reference frame.
- $R_M = (O_M,\overrightarrow{x_M},\overrightarrow{y_M},\overrightarrow{z_M})$: The manipulator reference frame.
- $R_E = (O_E,\overrightarrow{x_E},\overrightarrow{y_E},\overrightarrow{z_E})$: The end-effector reference frame.
- $R_C = (O_C,\overrightarrow{x_C},\overrightarrow{y_C},\overrightarrow{z_C})$: The camera reference frame.
- $R_I = (O_i,\overrightarrow{u_i},\overrightarrow{v_i})$: The image reference frame.
- $^{MT}_E$: The transformation matrix defining $R_E$ in $R_M$. It corresponds to the Kinematic Model of the manipulator (see 2.3.2).
- $^{MT}_M$: This matrix defines $R_M$ in $R_B$ (see 2.3.4).
- $^{MT}_B$: This matrix defines $R_B$ in $R_A$ (see 2.3.4).
- $^{MT}_E$: is the matrix defining $R_E$ in $R_A$ (see 2.3.4).
- $^{CT}_C$: The camera intrinsic parameters matrix (see 4.3.1).
- $^{CT}_A$: The camera extrinsic parameters matrix (see 4.3.1).
- $^{CT}_A$: The camera projection matrix (see 4.3.1).
- $^{CT}_C$: The Camera/Gripper transformation matrix (see 4.3.2).

### 2.2.2 Kinematic analysis of the ULM manipulator

The position coordinates and orientation angles of the end-effector are calculated in $R_M$ by (1) following the Modified Denavit-Hartenberg (MDH) representation (Khalil & Kleinfinger, 1986) where $^{MT}_2$ defines $R_2$ in $R_M$, $^{k-1T}_k$ ($k=3\ldots6$) defines $R_k$ in $R_{k-1}$ and $^{eT}_E$ defines $R_E$ in $R_e$.

$$^{MT}_E = ^{MT}_2 \ast ^{2T}_3 \ast ^{3T}_4 \ast ^{4T}_5 \ast ^{5T}_6 \ast ^{6T}_E$$  \hspace{1cm} (1)

$^{MT}_2$, $^{eT}_E$ and $^{k-1T}_k$ are given by (2) (Dombre & Khalil, 2007):

$$^{k-1T}_k = \begin{bmatrix}
\cos \theta_k & -\sin \theta_k & 0 & a_k \\
\sin \alpha_k \cdot \cos \theta_k & \cos \alpha_k \cdot \cos \theta_k & -\sin \theta_k & -d_k \cdot \sin \alpha_k \\
\sin \alpha_k \cdot \sin \theta_k & \sin \alpha_k \cdot \cos \theta_k & \cos \alpha_k & d_k \cdot \cos \alpha_k \\
0 & 0 & 0 & 1
\end{bmatrix}$$  \hspace{1cm} (2)
The different MDH parameters $\alpha_k$, $d_k$, $\theta_k$, $a_k$ and the joints limits of the ULM manipulator are given in Table 1 (Hentout et al., 2009a).

<table>
<thead>
<tr>
<th>$k$</th>
<th>$\alpha_k$ (rad)</th>
<th>$d_k$ (mm)</th>
<th>$\theta_k$</th>
<th>$a_k$ (mm)</th>
<th>$Q_{Min}(\text{o})$</th>
<th>$Q_{Max}(\text{o})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>290</td>
<td>$\theta_1$</td>
<td>0</td>
<td>-95</td>
<td>96</td>
</tr>
<tr>
<td>2</td>
<td>$\pi/2$</td>
<td>108.49</td>
<td>$\theta_2$</td>
<td>0</td>
<td>-24</td>
<td>88</td>
</tr>
<tr>
<td>3</td>
<td>$-\pi/2$</td>
<td>113</td>
<td>0</td>
<td>402</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>$\pi/2$</td>
<td>0</td>
<td>$\theta_3$</td>
<td>0</td>
<td>-2</td>
<td>160</td>
</tr>
<tr>
<td>5</td>
<td>$\pi/2$</td>
<td>389</td>
<td>$\theta_4$</td>
<td>0</td>
<td>-50</td>
<td>107</td>
</tr>
<tr>
<td>6</td>
<td>$-\pi/2$</td>
<td>0</td>
<td>$\theta_5$</td>
<td>0</td>
<td>-73</td>
<td>40</td>
</tr>
<tr>
<td>7</td>
<td>$\pi/2$</td>
<td>220</td>
<td>$\theta_6$</td>
<td>0</td>
<td>-91</td>
<td>91</td>
</tr>
</tbody>
</table>

Table 1. The MDH parameters and the joints limits of the ULM manipulator

### 2.2.3 Kinematic analysis of the mobile base

Assuming that the robot moves on the plane, the kinematic model of the non-holonomic mobile base can be decided, in $R_A$, by three parameters: $X_B$, $Y_B$ and $\theta_B$ the Cartesian coordinates and the orientation angle. During its motion, the mobile base calculates, by odometry, its position coordinates and orientation angle in real time as shown in (Hentout et al., 2009a).

### 2.2.4 Kinematic analysis of the mobile manipulator

It involves the interaction between the mobile base and the manipulator. The location of the end-effector is given in $R_A$ by:

$$A_{TE} = A_{TB} \cdot B_{TM} \cdot M_{TE}$$

(3)

$A_{TB}$ and $B_{TM}$ are given by (4) and (5) respectively. $(X_B, Y_B, Z_B)$ are the Cartesian coordinates of $O_B$ in $R_A$ and $(X_M, Y_M, Z_M)$ are the Cartesian coordinates of $O_M$ in $R_B$.

$$A_{TB} = \begin{bmatrix} \cos \theta_B & -\sin \theta_B & 0 & X_B \\ \sin \theta_B & \cos \theta_B & 0 & Y_B \\ 0 & 0 & 1 & Z_B \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(4)

$$B_{TM} = \begin{bmatrix} 1 & 0 & 0 & X_M \\ 0 & 1 & 0 & Y_M \\ 0 & 0 & 1 & Z_M \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(5)

For RobuTER/ULM, as shown in Fig. 3, $Z_B=120\text{mm}$, $X_M=30\text{mm}$, $Y_M=0\text{mm}$ and $Z_M=520\text{mm}$ (Hentout et al., 2009a).
3. Multi-agent control architecture of mobile manipulators

Fig. 4 shows the proposed multiagent control architecture of mobile manipulators as proposed in (Hentout et al., 2008). The architecture consists of six agents: Supervisory Agent (SA), Local Mobile Robot Agent (LMRA), Local Manipulator Robot Agent (LARA), Remote Mobile Robot Agent (RMRA) and Remote Manipulator Robot Agent (RARA). Each agent models a principal function of the mobile manipulator and manages a different sub-system. In addition, for each agent corresponds a mechanism connecting the four capacities Supervision, Perception, Decision and Action explained in more details in (Hentout et al., 2008).

The Supervision capacity is a virtual entity that select modules which result in the necessary behaviour facing a given situation.

The following are the basic functions of the architecture agents (Hentout et al., 2009b):

- **SA, Supervisory Agent**: SA receives the mission to be executed, decides on its feasibility according to the status and the availability (Perception + Decision) of the required equipments and resources of the robot (sensors, mobile base, manipulator, camera, etc.). If the mission is accepted, SA distributes it on on the corresponding agents for execution (Action).

- **LMRA, Local Mobile Robot Agent/LARA, Local Manipulator Robot Agent**: It receives the remote environment information of the mobile base/manipulator in order to build an up-to-date image on the environment where the robot evolves and, obtains feedback (reports) from RMRA/RARA on the execution of operations (Perception). In addition, the agent cooperates with the other agents (LARA/LMRA, VSA) in order to make a decision (Decision) according to the received information (sensors information, reports, etc.) and the status of the other agents of the architecture. At the end, it sends requests to RMRA/RARA for execution (Action).

- **VSA, Vision System Agent**: This agent observes the environment of the robot (Perception) by the vision system (the camera installed on the robot) and extracts useful and required information for the execution of the mission (Decision + Action) from captured images (images processing, localization and recognition of objects, etc.).

- **RMRA, Remote Mobile Robot Agent/RARA, Remote Manipulator Robot Agent**: This agent scans the various proprioceptif and exteroceptif sensors equipping the mobile base/manipulator (Perception) and sends the useful information to LMRA/LARA in order to maintain a correct representation of the environment. In addition, this agent ensures the local control of the mobile base/manipulator by sending instructions to its actuators and executing the multiple control strategies (navigation of the mobile base/motion of the manipulator) offered by LMRA/LARA (Decision + Action).
4. Implementation of the control architecture

The agents must be able to respond to asynchronous and external events, and to deal with requests, as soon as possible, according to the dynamics of the robot. Consequently, each agent is implemented as a set of concurrent communicating entities (a set of threads) executing autonomously and in parallel.

The agents communicate by sockets using TCP/IP protocol. Furthermore, semaphores are used to protect the access to the shared variables between the threads of the agent. P(Variable) to lock and V(Variable) to unlock the access to these variables. In addition, each agent has a Knowledge Base that describes its configuration. More details on the implementation of the multi-agent control architecture can be found in (Hentout et al., 2009c) (see Fig. 5 for the legend of the next figures of this section).

LMRA, LARA, VSA and SA agents are developed in Visual studio C# 2008 and installed on the off-board PC. RMRA and RARA agents are installed on the off-board PC. They are developed in C/C++ and SynDEx.
4.1 Supervisory Agent
The Supervisory Agent (Fig. 6) consists of the following threads:

- **Configuration**: This thread allows the operator to configure the agent (ID, port, competences, etc.) and to introduce all the information on the knowledge of the agent, and the partial knowledge about its acquaintances and its environment.

- **Communication**: it interfaces the agent with its acquaintances. It contains traditional communication functions as Connect, Disconnect, Send and Receive.

- **Supervision**: Its role is to activate threads that result in the necessary behavior (facing a given situation) and deactivate the others.

- **Human/Robot Interface**: it displays data of odometer, LMS, US, effort and joints positions sensors, the state of the gripper, CCD camera images, etc. In addition, it allows the operator to introduce the mission to be executed by the robot and to dialog with the control architecture.

- **Mission Decision**: This thread decides either the mission to be carried out by the robot is accepted or not. For this aim, it checks the availability and status of all the required resources. If the mission is accepted, it is sent to the other agents for execution. Otherwise, this thread informs the operator of its incapacity to accomplish this mission.

![Multithreading architecture of SA](image)

Fig. 6. Multithreading architecture of SA

4.2 Local Mobile/Manipulator Robot Agent
The Local Mobile/Manipulator Robot Agent (Fig. 7) consists of six threads:

- **Communication High-level**: communication with the other local agents.

- **Communication Low-Level**: it allows communicating with RMRA/RARA agent.

- **Configuration**.

- **Supervision**.

- **Sensors Processing**: it receives information on the environment of the remote robot. LMS, US and Odometer sensors for the mobile base; Positions sensors, Effort sensors and the State of the gripper for the manipulator.

- **Position Calculation**: for the Mobile Robot agent, this thread calculates the position of the mobile base on a plan relatively to any frame ($R_B$ or $R_A$).
For the Manipulator Robot agent, this thread tests either a given Cartesian position belongs of the current workspace of the manipulator or not. In addition, this thread calculates the Direct Kinematic Model (DKM) and the Inverse Kinematic Model (IKM) of the manipulator.

4.3 Vision System Agent
The images captured by the CCD camera of the robot must undergo several operations to extract useful information and to calculate the coordinates of the objects of the scene. The Vision System Agent (Fig. 8) is composed of the following threads:

- Communication.
- Supervision.
- Image Capturing: The CCD camera of the robot delivers continuous video images of the scene. Images are stocked in Image.
- Configuration: the configuration of this agent consists of the Camera calibration (see 4.3.1) and Camera/Gripper calibration (see 4.3.2). These parameters are stored in the knowledge base of the VSA agent.
- Image Processing: Firstly, the Median filter is applied to remove the noise. It consists of replacing the value of a pixel by the median value of its neighbor pixels. Secondly, the resulting image is binarised. The binarisation consists of transforming the image into another format with two colors only: black for the objects and white for the background. Thirdly, objects contours are detected. The contours consist of finding pixels in the image that correspond to changes of the luminance intensity. The algorithm of Canny (Canny, 1986) has been used. Finally, the forms recognition (characterization) consists of identifying the forms and classifying them in the corresponding category (triangles, rectangles, circles, etc.). To this aim, the Hough transformation (Duda & Hart, 1972) has been used. The result is saved in ProcessedImage.
- 2D Extraction: this thread calculates the 2D coordinates of the gravity center ($u_i, v_i$) of all the objects. The results are stocked in $UV$ array of $n$ elements.

Fig. 7. Multithreading architecture of LMRA/LARA

Fig. 8. Multithreading architecture of VSA
• **3D Extraction**: The aim of this thread is to compute the 3D real coordinates \((x, y, z)\) of the objects of the scene using the calibration parameters of the CCD camera. With a single camera, it is possible to estimate only two coordinates \((y, z)\). Thus, to get the other one \((x)\), another measurement system is needed. The used approach is that developed in (Bouzouia & Rahiche, 2009). It is as follows:
  - From the captured and processed image, the \((y, z)\) coordinates of the selected object are calculated by using the camera model obtained by the calibration process.
  - The measure representing the other component \((x)\) is obtained from the LMS sensor.

![Multithreading architecture of VSA](fig8.png)

**Fig. 8. Multithreading architecture of VSA**

The *acquisition, Filtering, Segmentation* and *Characterization* steps are illustrated in Fig. 9 (Bouzouia & Rahiche, 2009).

![Image processing process](fig9.png)

**Fig. 9. Result of the image processing process**
### 4.3.1 Camera calibration

Camera calibration consists in determining the $3 \times 4$ transformation matrix $^iT_A$ that maps a 3D coordinates of a point in the space $P \begin{bmatrix} x \\ y \\ z \end{bmatrix}$ expressed in $R_A$, using a calibration grid, onto its 2D image projection whose coordinates $m \begin{bmatrix} u \\ v \end{bmatrix}$ are expressed in pixel in $R_I$ (Tellel et al., 2003). The relation between $P$ and $m$ is given by (6) where $s$ is an arbitrary scale factor (Muis & Ohnishi, 2005):

$$
\begin{bmatrix}
    su \\
    sv \\
    s\bar{v}
\end{bmatrix}
= 
^iT_A \begin{bmatrix}
    x \\
    y \\
    z
\end{bmatrix}
\left(\begin{array}{c}
    \bar{u} \\
    \bar{v}
\end{array}\right)
$$

(6)

For the camera calibration, the method proposed in (Bénallal, 2002) is adopted. It consists of solving (7) with $n \geq 6$ (Hartley & Zisserman, 2001) and $m_{11}, m_{12}, ..., m_{33}$ are the elements of the matrix $^iT_A$.

$$
\begin{bmatrix}
    x_1 & y_1 & z_1 & 1 & 0 & 0 & 0 & 0 & -u_1 \cdot x_1 & -u_1 \cdot y_1 & -u_1 \cdot z_1 \\
    0 & 0 & 0 & 0 & x_1 & y_1 & z_1 & 1 & -v_1 \cdot x_1 & -v_1 \cdot y_1 & -v_1 \cdot z_1 \\
    x_2 & y_2 & z_2 & 1 & 0 & 0 & 0 & 0 & -u_2 \cdot x_2 & -u_2 \cdot y_2 & -u_2 \cdot z_2 \\
    0 & 0 & 0 & 0 & x_2 & y_2 & z_2 & 1 & -v_2 \cdot x_2 & -v_2 \cdot y_2 & -v_2 \cdot z_2 \\
    & & & & & . & & & . & . & . \\
    & & & & & . & & & . & . & . \\
    x_n & y_n & z_n & 1 & 0 & 0 & 0 & 0 & -u_n \cdot x_n & -u_n \cdot y_n & -u_n \cdot z_n \\
    0 & 0 & 0 & 0 & x_n & y_n & z_n & 1 & -v_n \cdot x_n & -v_n \cdot y_n & -v_n \cdot z_n
\end{bmatrix}
\begin{bmatrix}
    m_{11} \\
    m_{12} \\
    m_{13} \\
    m_{14} \\
    m_{21} \\
    m_{22} \\
    m_{23} \\
    m_{24} \\
    m_{31} \\
    m_{32} \\
    m_{33}
\end{bmatrix}
= 
\begin{bmatrix}
    u_1 \\
    v_1 \\
    v_2 \\
    v_2 \\
    u_n \\
    u_n
\end{bmatrix}
$$

(7)

The obtained matrix $M$ is given by (8) (Hentout et al., 2009d):

$$
M \approx \begin{pmatrix}
    -0.2057 & 0.8110 & -0.079 & 69.4449 \\
    0.6498 & 0.1155 & -0.434 & 139.8394 \\
    0.0005 & 0.0002 & 0.0006 & 1
\end{pmatrix}
$$

(8)

### 4.3.2 Camera/Gripper Calibration

Camera/Gripper calibration consists of finding the matrix $^ET_c$ defining $R_C$ in $R_E$.

Let $^CT_{A1}$ and $^CT_{A2}$ be the transformation matrices defining a first and a second position of the camera in $R_A$. Let $^MT_E1$ and $^MT_E2$ be the transformation matrices defining the two positions of the end-effector in $R_M$ corresponding to the first and second position of the camera.

To find the Camera/Gripper calibration matrix $^ET_C$, the method developed in (Tsai & Lenz, 1989) is chosen. It is based on the Least squares method and consists of solving (9) where:

- $A = (^CT_{A2})^T(^CT_{A1})^{-1}$: The measurable transformation matrix of the camera from its first to its second location (relative camera motion).
- $B = (^MT_E2)^T(^MT_E1)$: The measurable transformation matrix of the gripper from its first to its second location (relative robot gripper motion).

$$
A \cdot ^ET_C = ^ET_C \cdot B
$$

(9)

The obtained matrix $^ET_C$ is given by (10) (Hentout et al., 2009e):
For the camera calibration, the method proposed in (Bénallal, 2002) is adopted. It consists of camera in matrix 2D image projection whose coordinates Camera/Gripper calibration consists of finding the matrix. The obtained matrix is given by (10) (Hentout et al., 2009e):

\[
E_{TC} = \begin{pmatrix}
0.7322 & -0.6573 & 0.1787 & -183.1608 \\
0.6595 & 0.7497 & 0.0555 & -69.4161 \\
-0.1704 & 0.0772 & 0.9823 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\]

\[\text{(10)}\]

4.4 Remote Mobile Robot Agent
The Remote Mobile Robot Agent, given by Fig. 10, consists of seven threads:
- Configuration.
- Communication.
- Supervision.
- Reading LMS Sensors: it scans continuously the serial port of the LMS sensors and stores data in ValuesLMS of 181 items.
- Reading US Sensors: This thread, in its turn, reads constantly the US sensors and stores US data in ValuesUS of 24 items.
- Odometry: This thread reads the values of the incremental encoders (E_R, E_L), installed on the driven wheels of the mobile base, and calculates its current position and orientation angle (New_X, New_Y, New_θ) as shown in (Hentout et al., 2009a).
- Navigation: it consists of the main role of this agent. It uses data of all the other threads. Navigation calculates velocities (Spd_R, Spd_L) to be sent to the actuators of the mobile base in order to move to a Target position given by (X_{Target}, Y_{Target}, θ_{Target}) while avoiding possible obstacles.

Fig. 10. Multithreading architecture of RMRA
4.5 Remote Manipulator Robot Agent
The Remote Manipulator Robot Agent (Fig. 11) is composed, in its turn, of seven threads:
- Communication.
- Configuration.
- Supervision.
- Reading Positions Sensors: it reads the incremental positions sensors (PcIn) installed on each articulation of the manipulator and saves data in ValuesPos of 6 elements.
- Reading Effort Sensors/State Gripper: this thread, in its turn, reads the effort sensor data (Gripper) and the state of the gripper (Opened or Closed). This thread stores read data in ValuesGripper of 7 items (Forces (Fx, Fy, Fz), Torques (Tx, Ty, Tz), State of the Gripper).
- Open/Close gripper: it opens or closes the gripper.
- Movement: This thread consists of the main role of this agent. It calculates orders (PcOut) to be sent to the actuators of the manipulator in order to move to a Target position given by (Qi ...Q6) with a given velocity Vi (i=1...6) for each joint.

![Multithreading architecture of RARA](image)

Fig. 11. Multithreading architecture of RARA

5. Experimental part
The core thinking of modeling and controlling mobile manipulators using a multi-agent system is that of realizing cooperation between the manipulator, the mobile base and the sensors system. In order to show the validity of the implementation of the SA, LMRA, LARA, VSA, RMRA and RARA agents, two different missions are considered in this section. For the envisaged experiments, all the positions and orientations are given in RA. In addition, two cases are distinguished (Fig. 12):
• The Target belongs of the current workspace of the robot: this task requires only the motion of the manipulator.

• The Target is outside the current workspace of the robot: In this case, the mobile base moves until the Target is within the new workspace of the manipulator. Then, the robot manipulates the Target with its end-effector.

Fig. 12. The two possible cases for a given Target

5.1. Following a predefined operational trajectory
This experiment presents trajectory planning and control for mobile manipulators. The end-effector of the robot has to be, as near as possible, from a predefined operational trajectory (given by a set of Cartesian coordinates \((X, Y, Z)\)) while the non-holonomic mobile base has to avoid the obstacles present in the environment.

5.1.1 Straight-line following
In this experiment, as shown in Fig. 13, the operational trajectory to be followed by the end-effector of the robot consists of a straight-line connecting an initial position \(P_i(X_i, Y_i, Z_i)\) to a final position \(P_f(X_f, Y_f, Z_f)\) (Hentout et al., 2009e).

To compute the imposed positions \((\text{Targets})\) to be reached by the end-effector, let’s consider:

- \([P_i, P_f]\): the segment connecting \(P_i\) to \(P_f\).
- \(p(X_p, Y_p, Z_p)\): the current position coordinates of the end-effector.
- \(h(X_h, Y_h, Z_h)\): the projection of \(p\) on the segment \([P_i, P_f]\).
- \(\text{New}_X, \text{New}_Y, \text{New}_\theta\): the current position coordinates and orientation angle of the mobile base.
- \(\text{Position}_{\text{Init}}(X_{\text{Init}}, Y_{\text{Init}}, \theta_{\text{Init}})\): the initial position coordinates and orientation angle of the mobile base.
- \(\text{Position}_{\text{Fin}}(X_{\text{Fin}}, Y_{\text{Fin}}, \theta_{\text{Fin}})\): the final position coordinates and orientation angle of the mobile base.
- \(m(X_m, Y_m, Z_m)\): a point in the space.

\[
m \in [P_i, P_f] \Rightarrow \begin{cases} 
X_m = t \cdot (X_f - X_i) + X_i \\
Y_m = t \cdot (Y_f - Y_i) + Y_i \\
Z_m = t \cdot (Z_f - Z_i) + Z_i 
\end{cases} \quad (11)
\]

\(^{\rightarrow}\) is orthogonal to \(\rightarrow P_f\) so:

\[
(X_h - X_p) \cdot (X_f - X_i) + (Y_h - Y_p) \cdot (Y_f - Y_i) + (Z_h - Z_p) \cdot (Z_f - Z_i) = 0 \quad (12)
\]
From (11) and (12), the position of \( h \) (the next Target to be reached by the end-effector of the robot) on the segment \([P_i, P_f]\) is given by (13):

\[
\begin{align*}
X_h &= t \cdot (X_f - X_i) + X_i \\
Y_h &= t \cdot (Y_f - Y_i) + Y_i \\
Z_h &= t \cdot (Z_f - Z_i) + Z_i \\
t &= \frac{t \cdot (X_f - X_i) \cdot (X_p - X_i) + (Y_f - Y_i) \cdot (Y_p - Y_i) + (Z_f - Z_i) \cdot (Z_p - Z_i)}{(X_f - X_i)^2 + (Y_f - Y_i)^2 + (Z_f - Z_i)^2}
\end{align*}
\]  

(13)

The positioning error of the end-effector is calculated by (14):

\[
Err = \sqrt{(X_h - X_p)^2 + (Y_h - Y_p)^2 + (Z_h - Z_p)^2}
\]  

(14)

The straight-line following algorithm for LARA is given as follows:

```plaintext
Straight_Line_Following_LARA(\(P_i, P_f\)) {
    if(\(P_i\) \notin Current workspace of the manipulator)
        wait for message (Mobile base in \(Position_{init}\)) from LMRA; 
    Generate the possible orientations for \(P_i\) using the IKM; 
    \(Q_i(i=1...6)\) = Choose the best configuration; 
    Send \(Move(Q_{i(i=1...6)})\) to RARA; 
    while (\(P_f \neq P_f\)) {
        Receive (\(New_X, New_Y, New_\theta\)) from LMRA; 
        Calculate \(P\) in \(R_j\) according to (\(New_X, New_Y, New_\theta\)); 
        if (\(P \notin [P_i, P_f]\)) {
            Calculate \(b\), the projection of \(P\) on \([P_i, P_f]\); 
            Generate the possible orientations for \(b\) using the IKM; 
            \(Q_i(i=1...6)\) = Choose the best configuration; 
            Send \(Move(Q_{i(i=1...6)})\) to RARA; 
        }
    }
}
```

The straight-line following algorithm for LMRA is given here below:

```plaintext
Straight_Line_Following_LMRA(\(P_i, P_f\)) {
```

---

Fig. 13. The straight-line following mission and its parameters

---

www.intechopen.com
Calculate Position\textsubscript{Init} corresponding to $P_1$;
Send Move(Position\textsubscript{Init}) to RMR\textsubscript{A};
wait for (Mobile base in Position\textsubscript{Init}) from RMR\textsubscript{A};
Send (Mobile base in Position\textsubscript{Init}) to LARA;
Calculate Position\textsubscript{Fin} corresponding to $P_6$;
Send Move(Position\textsubscript{Init}) to RMR\textsubscript{A};
while ((New_X, New_Y, New_\theta)! = Position\textsubscript{Fin}) {
    Receive (New_X, New_Y, New_\theta) from RMR\textsubscript{A};
    Send (New_X, New_Y, New_\theta) to LARA;
}

These two previous algorithms are executed in parallel on the off-board PC by the corresponding agents. In addition, LMR\textsubscript{A} and LARA send requests and receive sensors data and reports from the corresponding agent (RMR\textsubscript{A} and RARA). At the same time, RMR\textsubscript{A} and RARA move towards the received positions: Position\textsubscript{Fin} for the mobile base and (Q\textsubscript{1} … Q\textsubscript{6}) for the manipulator.

5.1.2 Experimental result
The straight-line following algorithms proposed previously for LMR\textsubscript{A} and LARA are implemented to the RobuSTER/JLM. (13) is used to generate the Target positions so that the end-effector of the mobile manipulator follows the desired line (Hentout et al., 2009e).
For this experiment $P_1(X_1, Y_1, Z_1) = (-691.72\text{mm}, -108.49\text{mm}, 1128.62\text{mm})$ and $P_6(X_6, Y_6, Z_6) = (-2408.17\text{mm}, -108.49\text{mm}, 1472.30\text{mm})$. Therefore, the operational trajectory consists of a straight-line with a slope of about 350mm (343.68mm).
The initial posture of the mobile base and that of the end-effector corresponding to $P_1$ is Target\textsubscript{Init}(X_\text{Init}, Y_\text{Init}, \theta_\text{Init}, X_\text{ElInit}, Y_\text{ElInit}, Z_\text{ElInit}, \psi_\text{ElInit}, \phi_\text{ElInit}) = (0\text{mm}, 0\text{mm}, 0^\circ, -691.72\text{mm}, -108.49\text{mm}, 1128.62\text{mm}, -90^\circ, -90^\circ, -90^\circ). For this initial position, the initial joint angles (Q\textsubscript{1Init}, Q\textsubscript{2Init}, Q\textsubscript{3Init}, Q\textsubscript{4Init}, Q\textsubscript{5Init}, Q\textsubscript{6Init}) = (0^\circ, 60^\circ, 0^\circ, 0^\circ, 32^\circ, 0^\circ).The final position of the mobile base and that of the end-effector corresponding to $P_6$ is Target\textsubscript{Fin}(X_\text{BFin}, Y_\text{BFin}, \theta_\text{BFin}, X_\text{EFin}, Y_\text{EFin}, Z_\text{EFin}, \psi_\text{EFin}, \phi_\text{EFin}) = (-1920\text{mm}, 2\text{mm}, 15^\circ, -2408.17\text{mm}, -108.49\text{mm}, 1472.30\text{mm}, 0^\circ, -90^\circ, 0^\circ). For this final position, the final joint angles (Q\textsubscript{1Fin}, Q\textsubscript{2Fin}, Q\textsubscript{3Fin}, Q\textsubscript{4Fin}, Q\textsubscript{5Fin}, Q\textsubscript{6Fin}) = (37^\circ, 52^\circ, 61^\circ, 73^\circ, -57^\circ, 28^\circ).
Two cases are tested for this example (Hentout et al., 2009e):
- The environment of the robot is free (no obstacles are considered). The motion of the mobile base consists also of a straight-line connecting Position\textsubscript{Init} to Position\textsubscript{Fin}. For this case, the robot follows perfectly the imposed straight-line.
- The second case is more difficult. The non-holonomic mobile base has to avoid an obstacle present in the environment while the end-effector has to be always at the desired configuration (on the straight-line). For the second case of this experiment, the operational trajectory followed by the end-effector and the imposed trajectory for the end-effector are shown on Fig. 14.
Fig. 14. Imposed operational trajectory and the real trajectory of the end-effector
The real joints variations rather than the desired trajectory for some joints (1, 2, 3 and 5 respectively) are shown on Fig. 15.

Fig. 15. Joints variations and desired trajectories of some joints

Fig.16 shows the trajectory followed by the mobile base and the avoidance of the obstacle.

Fig. 16. Real trajectory followed by the mobile base
5.1.3 Discussion of results
Figs. 14, 15 and 16 showed the operational trajectory of the end-effector, the variations of some joints of the manipulator and the motion of the mobile base respectively. The mission took about 160 seconds.

The maximum positioning error calculated by (14) is 24.43mm while the average error is 3.41mm. The errors show that it is difficult to follow the desired straight-line.

The first reason for this error is the initial positioning error of the mobile base (\(\text{Position}_{\text{Init}}\)). It causes straying from the initial position for the end-effector in the trajectory. To solve this problem, the mobile manipulator must absorb this error by the motion of its manipulator. Secondly, an estimated positioning error of the mobile base, calculated by odometry (\(\text{New}_X, \text{New}_Y, \text{New}_\theta\)), during its motion effects the tip position of the end-effector directly. To absorb this error, the manipulator should move quickly to adjust itself when the error is detected. Finally, the low velocity of the manipulator’s motion during the motion of the mobile base causes a delay in the positioning of the end-effector. This problem can be solved by incrementing the velocity of the manipulator according to that of the mobile base.

5.2 Aligning the end-effector of the robot to different objects by using the eye-in-hand camera and the LMS sensor
A position-based servoing control of mobile manipulators by using the eye-in-hand camera and the LMS sensor is considered. The working mission is to reach different positions (corresponding to various objects) by the end-effector of the robot.

To reach an object, it is necessary to capture an image of this object. VSA sends, for this aim, a request \textit{Move Gripper (Position)} to LARA in order to position the manipulator. \textit{Position} is read from the \textit{Knowledge Base} of VSA. After the positioning of the manipulator, VSA captures an image and carries out the necessary processing to extract the 2D coordinates \((u, v)\) of the gravity center of the object in the image. At the end, VSA extract \((y, z)\) coordinates and sends them to LARA. LARA always needs the \((x)\) coordinate. To this aim, a request is sent, in parallel, to LMRA (\textit{Read LMS}) which transmits it to RMRA. Receiving this request, RMRA send \textit{LMSValues} data to LMRA. This latter selects the minimum value from the 60\textsuperscript{th} element to 120\textsuperscript{th} element (corresponding to 60° to 120°). This value corresponds to the \((x)\) coordinate. It is sent to LMRA which has now \((x, y, z)\) coordinates of the position to be reached.

5.2.1 Experimental result
For this experiment, as shown in Fig. 18, the initial posture of the mobile base and that of the end-effector is \(\text{Target}_{\text{Init}}(X_{\text{Init}}, Y_{\text{Init}}, \theta_{\text{Init}}, X_{\text{Elinit}}, Y_{\text{Elinit}}, Z_{\text{Elinit}}, \varphi_{\text{Elinit}}, \vartheta_{\text{Elinit}, \varphi_{\text{Elinit}}}) = (0\text{mm}, 0\text{mm}, 0°, -546.62\text{mm}, -110.36\text{mm}, 1200.73\text{mm}, -90°, -90°, -90°)\). The position to be reached by the end-effector of the robot are at a distance \(x=-2470\). For this initial position, the initial joint angles \((Q_{1\text{Init}}, Q_{2\text{Init}}, Q_{3\text{Init}}, Q_{4\text{Init}}, Q_{5\text{Init}}, Q_{6\text{Init}}) = (0°, 87°, 0°, 0°, 5°, 0°)\). Table 2 shows the different parameters of this experiment.
The joints angles and the corresponding end-effector coordinates are given in Table 3.

<table>
<thead>
<tr>
<th>Image Coordinates $(u, v)$ (pixel)</th>
<th>Real Coordinates $(x, y, z)$ (mm)</th>
<th>LMS value $(x)$ (mm)</th>
<th>Calculated Coordinates $(x, y, z)$ (mm)</th>
<th>Mobile Base coordinates $(X_B, Y_B, \theta_B)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>166, 176</td>
<td>-2470, -63, 1325</td>
<td>2470</td>
<td>-2470, -63.01, 1328.56</td>
<td>-1670mm, 0mm, 0°</td>
</tr>
<tr>
<td>182, 228</td>
<td>-2470, 52, 1295</td>
<td>2470</td>
<td>-2470, 46.40, 1294.85</td>
<td>-1670mm, 0mm, 0°</td>
</tr>
<tr>
<td>228, 178</td>
<td>-2470, -58, 1200</td>
<td>2470</td>
<td>-2470, -58.93, 1197.94</td>
<td>-1670mm, 0mm, 0°</td>
</tr>
<tr>
<td>234, 227</td>
<td>-2470, 47, 1185</td>
<td>2470</td>
<td>-2470, 44.25, 1185.30</td>
<td>-1670mm, 0mm, 0°</td>
</tr>
</tbody>
</table>

Table 2. Different parameters of the experiment

The joints angles and the corresponding end-effector coordinates are given in Table 3.

<table>
<thead>
<tr>
<th>Target</th>
<th>Joints angles $Q_i$ $(i=1..6)$ ($^\circ$)</th>
<th>End-effector coordinates $(X_E, Y_E, Z_E, \phi_E, \theta_E, \psi_E)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Target_1$</td>
<td>(5, 49, 63, -13, -22, -78)</td>
<td>-2466.52mm, -60.45mm, 1335.20mm, -90°, -90°, 180°</td>
</tr>
<tr>
<td>$Target_2$</td>
<td>(16, 51, 54, -48, -22, -44)</td>
<td>-2469.86mm, 44.17mm, 1293.73mm, -90°, -90°, 180°</td>
</tr>
<tr>
<td>$Target_3$</td>
<td>(5, 53, 37, 89, 5, 1)</td>
<td>-2468.08mm, -60.95mm, 1199.39mm, -90°, -90°, 0°</td>
</tr>
<tr>
<td>$Target_4$</td>
<td>(16, 53, 35, 83, 16, 7)</td>
<td>-2469.41mm, 45.15mm, 1185.48mm, -90°, -90°, 0°</td>
</tr>
</tbody>
</table>

Table 3. Joints angles and end-effector postures for the different Targets

The following snapshots (Fig. 18) show the obtained result (Bouzouia & Rahiche, 2009):

![Targets](image1)
![Distance between the robot and the Targets](image2)
![Initial position](image3)
![The four positions to be reached](image4)

Fig. 18. Position-based servoing control by using the camera and the LMS sensor of RobuTER/ULM

5.2.2 Discussion of results

The VSA agent uses the eye-in-hand camera to extract the two last coordinates $(y$ and $z$) of the object to be manipulated by the robot. The LMRA and the RMRA agent use the LMS sensor to obtain the first coordinate $(x)$.

The maximum 3D reconstruction error calculated by (14) is 5.60mm while the minimum error is 2.26mm. These errors are acceptable. They are due to the weak precision of the measured real values, to the low rigidity of the manipulator, to the accumulation errors of the calibration processand to the feeble precision of the LMS sensor ($\pm 15$mm).

The maximum positioning error is 11.07mm while the minimum error is 2.00mm. The errors are principally due to the error in the estimated positioning of the mobile base, calculated by...
odometry \((\text{New}_X, \text{New}_Y, \text{New}_\theta)\), during its motion. These errors are also due the accumulated errors in the \(IKM\) of the manipulator.

6. Conclusions and future works

This chapter presented a multi-agent control architecture of mobile manipulators. The architecture consists of six agents: \textit{Supervisory Agent} (SA), \textit{Local Mobile Robot Agent} (LMRA), \textit{Local Manipulator Robot Agent} (LARA), \textit{Vision System Agent} (VSA), \textit{Remote Mobile Robot Agent} (RMRA) and \textit{Remote Manipulator Robot Agent} (RARA). The first four agents are installed on an off-board PC while the two other agents are installed on the on-board PC of the robot. The controller was applied successfully to follow a predefined straight-line operational trajectory by the end-effector of a differentially-driven \textit{RobuTER/ULM} mobile manipulator while considering obstacles in its environment. The controller was shown to be relatively effective when the robot moves with small velocities.

To realize the operational trajectory following, one of the biggest problems is that an accumulated error of the estimated position of the mobile base affects the position accuracy of the end-effector. Therefore, the manipulator should have a capability to adjust its position when the mobile base detects positioning errors.

The results obtained the position-based servoing control of the robot by using the eye-in-hand camera and the \textit{LMS} sensor are satisfactory since the positioning error of the end-effector is less than 15mm. The calculation of the 3D coordinates is based on the eye-in-hand camera (for \((y, z)\) coordinates) and on the \textit{LMS} sensor (for \((x)\) coordinate).

In future works, the performances and the robustness of the implemented agents of the control architecture should be shown and discussed through examples of other types of trajectories (circular, etc.). Furthermore, and especially for the VSA agent, a moving target tracking problem should be performed. In addition, the real time constraint for the VSA agent will be verified and discussed. Another extension of this work is to introduce a virtual reality system (a graphic simulator) to give more effective action for the developed architecture.

7. References


Robot manipulators are developing more in the direction of industrial robots than of human workers. Recently, the applications of robot manipulators are spreading their focus, for example Da Vinci as a medical robot, ASIMO as a humanoid robot and so on. There are many research topics within the field of robot manipulators, e.g. motion planning, cooperation with a human, and fusion with external sensors like vision, haptic and force, etc. Moreover, these include both technical problems in the industry and theoretical problems in the academic fields. This book is a collection of papers presenting the latest research issues from around the world.

How to reference
In order to correctly reference this scholarly work, feel free to copy and paste the following:
