Preprints of the 19th World Congress
The International Federation of Automatic Control
Cape Town, South Africa. August 24-29, 2014

Multi-agent Control Approach for Autonomous Mobile Manipulators: Simulation Results on *RobuTER/ULM*

Abdelfetah HENTOUT, Mohamed Ayoub MESSOUS, Brahim BOUZOUIA

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

ahentout@cdta.dz, mmessous@cdta.dz, bbouzouia@cdta.dz

Abstract: This article presents a multi-agent approach for controlling autonomous mobile manipulators. The proposed approach assigns a hybrid agent (*Mobile base* agent) for the control of the mobile base, a reactive agent (*Joint* agent) to each degree-of-freedom (*dof*) of the manipulator, and a *Supervisory* agent to assure coordination and to synchronize the work of the whole agents of the system.

The initial simulation results, obtained via different positioning tasks on *RobuTER/ULM* with and without considering breakdowns, show that the main advantage of such an approach is that it pledges a fault-tolerant response to various types of breakdowns without adding any specific dysfunction treatment.

Keywords: Multi-agent control approach, Autonomous mobile manipulators, Simulation, *RobuTER/ULM*.

1. INTRODUCTION

A mobile manipulator consists of a mobile base on which is mounted upon one or more manipulators. The robot can accomplish most common tasks of robotics that require locomotion and manipulation capabilities (Nebot et al., 2004). Such autonomous robots must perform scheduled tasks in complex, unknown and changing environments with sensing, perceptive, knowledge-acquisition, learning, inference, decision-making and acting ability (Wen et al., 2004) by using only their limited physical and computational resources with a reduced human intervention (Medeiros, 1998). To this aim, different control approaches for such robots have been proposed in the literature. They can be, mainly, divided into two different classes (*i*) *Traditional/Classical approaches* and (*ii*) *Multi-agent approaches* (Hentout et al., 2013).

The first class of approaches is based on the study of mathematical models of both manipulator and mobile base (Bayle et al., 2003) (Nikoobin et al., 2009). Controlling such a robot consists of computing the motion of these two sub-systems. For this aim, the study of *Direct* (*DKM*) and *Inverse* (*IKM*) *Kinematic Models* is required (Joukhadar, 1997). Classical approaches produce accurate results and offer a fairly exact control for repetitive tasks in controlled environments (industrial robotics, etc.). In such a case, when the robot is required to repeat a trajectory thousands of times, complicated computation of these models is done, generally, off-line with the ability to optimize time and/or energy. The methods used for computing *DKM* represent generic rules, whereas *IKM* are constructed according to the structure of the robot. Moreover, these models don't tolerate any changes in the mechanical structure without adding a specific mode for failures treatment (joint malfunction, etc.). Finally, classical approaches have the weakness of the important computing time depending on the high number of dof, especially in frequently-changed, unknown and evolutionary environments where operate most of robots.

Multi-agent approaches (*MAS*) propose a decomposition of the robot control into a set of distinct agents (Duhaut, 1999) (Erden et al., 2004). Every agent tries to align the position of the end-effector with that of the target, without prior knowledge of the actions and positions of the other agents. By acting independently, they try to do the same job and a global behavior can emerge, consequently, from all these local agents for satisfying the desired objective. *MAS* approaches offer simple solutions and benefit of all the advantages of distributed problem solving. Here, the system is considered as a compound of simpler modules, which gave an easier way to design the whole system. In addition, the need for massy mathematical models, *IKM* and differential-equation solvers is overcome (Duhaut, 1999). Therefore, there is a considerable decrease in design effort and computation time compared to classical approaches. Finally, with such a usage of *MAS*, the control system is more flexible to be applied to any robot (mobile, manipulator and mobile manipulator robots).

The paper is organized as follows. Section two describes the proposed multi-agent control approach. The experimental robot, the validation scenarios and the obtained simulation results are presented and discussed in section three. Section four concludes the paper and draws-up future works.

2. MULTI-AGENT CONTROL APPROACH

Fig. 1 presents a global scheme of the control approach for *RobuTER/ULM*. The robot consists of a six-dof manipulator installed on a mobile base. In this case, the multi-agent system involves a set of eight agents:

- Six reactive *Joint agents* are assigned to control the six-dof manipulator.
- One hybrid *Mobile base agent* to control the mobile base of the robot.
- One hybrid *Supervisory agent* that coordinates between the whole of the precedent agents.

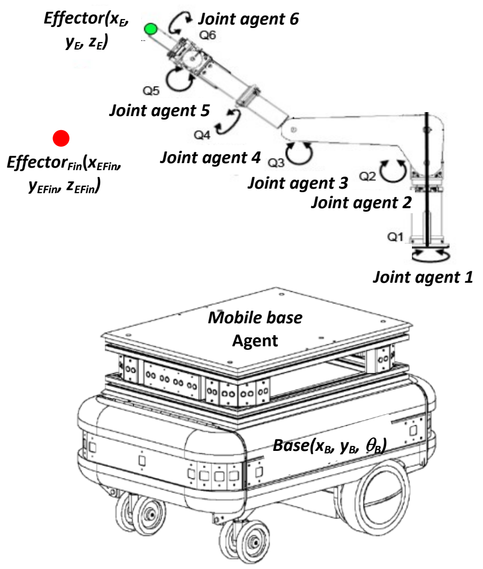

Effector(x_E, y_E, z_E)
Joint agent 6
Q6
Q5
Joint agent 5
Q4
Effector$_{Fin}$(x_{EFin}, y_{EFin}, z_{EFin})
Joint agent 4
Q3
Q2
Joint agent 3
Joint agent 2
Q1
Joint agent 1
Mobile base Agent
Base(x_B, y_B, θ_B)

Fig. 1. Global control system of six-dof mobile manipulator

The objective of each agent of the system is to optimize an objective function f_{Obj} which depends on *Configuration*, *Base* and *Effector$_{Fin}$*. f_{Obj} is given by (1):

$$f_{obj} = F\_Objective(Configuration, Base, Effector_{Fin}) \qquad (1)$$

In this present work, the objective is to reduce the distance between *Effector* and *Effector$_{Fin}$*. This distance is computed as follows:

$$Distance = \sqrt{(x_{EFin} - x_E)^2 + (y_{EFin} - y_E)^2 + (z_{EFin} - z_E)^2} \qquad (2)$$

The computation of the *DKM* of the robot function of *Base(x_B, y_B, θ_B)* and *Configuration*(q_1, ..., q_{dof}) is done by using equation (3):

$$Effector=DKM(Base, Configuration) \qquad (3)$$

Each agent of the system (*Joints agents*, *Mobile base agent*) receives, from the *Supervisory agent*, the initial situations of the robot (*Configuration$_{Init}$* and *Base$_{Init}$*), the imposed final situation of the end-effector (*Effector$_{Fin}$*) and the initial objective function (ξ_{Init}).

2.1 Manipulator agents

Fig. 2 illustrates the two possible elementary movements of each joint controlled by a *Joint agent*:

- The agent makes a virtual rotation in the positive direction (*MoveUp*) with a *Joint footstep* and computes the objective function value (*ξ_I_Up*) between *Effector* and *Effector$_{Fin}$*. *ξ_I_Up* depends on the new configuration of the manipulator (*Configuration_I_Up*) and *Base*.
- The agent will repeat the previous actions while changing its rotation into the negative direction (*MoveDown*, *Configuration_I_Down*, *Joint footstep, ξ_I_Down*).

The *Joint agent* will choose the new configuration of the manipulator (*Configuration_I_New*) corresponding to the configuration minimizing the distance between *Effector* and *Effector$_{Fin}$* (*ξ_I_New*). The best choice could be to remain in its current configuration without moving in case where (*ξ<ξ_I_New*). At the end, the *Joint agent* sends its best choice satisfying its local objective (*Configuration_I_New* and *ξ_I_New*) to the *Supervisory agent*.

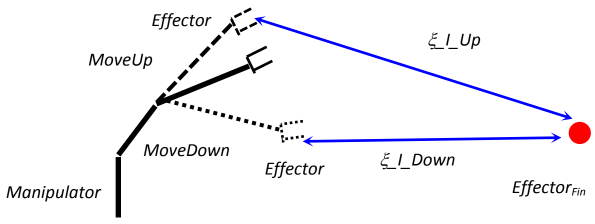


Fig. 2. Elementary movements of a *Joint agent*

The diagram of Fig. 3 explains the behavior of a *Joint agent*.

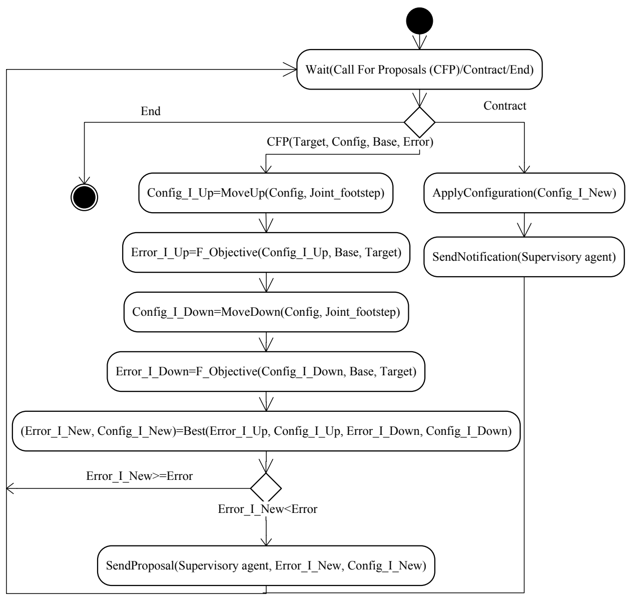

Wait(Call For Proposals (CFP)/Contract/End)
End
Contract
CFP(Target, Config, Base, Error)
Config_I_Up=MoveUp(Config, Joint_footstep)
ApplyConfiguration(Config_I_New)
Error_I_Up=F_Objective(Config_I_Up, Base, Target)
SendNotification(Supervisory agent)
Config_I_Down=MoveDown(Config, Joint_footstep)
Error_I_Down=F_Objective(Config_I_Down, Base, Target)
(Error_I_New, Config_I_New)=Best(Error_I_Up, Config_I_Up, Error_I_Down, Config_I_Down)
Error_I_New>=Error
Error_I_New<Error
SendProposal(Supervisory agent, Error_I_New, Config_I_New)

Fig. 3. Behavior diagram of a *joint I agent* of the manipulator

2.2 Mobile base agent

In this paper, the environment where evolves the mobile base is considered free of obstacles. Fig. 4 illustrates the four considered elementary movements for the *Mobile base agent*:

- The agent makes a virtual forward movement (*MoveForward*) with a *Base Translation footstep* and computes the new value of the objective function (*ξ_F*) between *Effector* and *Effector$_{Fin}$*. This distance depends on the new situation of the mobile base (*Base_F*) and the current configuration of the manipulator (*Configuration*).

- The agent will repeat the previous actions while changing every time the direction of its movement (*MoveBackward*: *Base_B, Base Translation footstep*, *ξ_B*), (*TurnRight*: *Base_R, Base Rotation footstep*, *ξ_R*) and (*TurnLeft*: *Base_L, Base Rotation footstep*, *ξ_L*).

The *Mobile base agent* will choose the new situation of the mobile base (*Base_New*) that corresponds to the situation minimizing the distance (*ξ_Base_New*) between *Effector* and *Effector*$_{Fin}$. This choice could be the remaining in its current situation without moving if (*ξ<ξ_Base_New*). At the end, the agent sends its best choice (*ξ_Base_New* and *Base_New*) to the *Supervisory agent*.

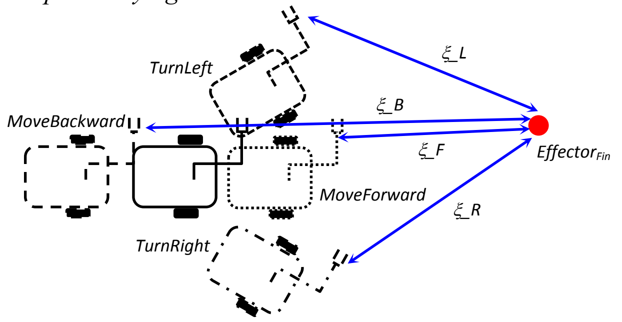


Fig. 4. Elementary movements of the *Mobile base agent*

The diagram of Fig. 5 explains the behavior of the *Mobile base agent*.

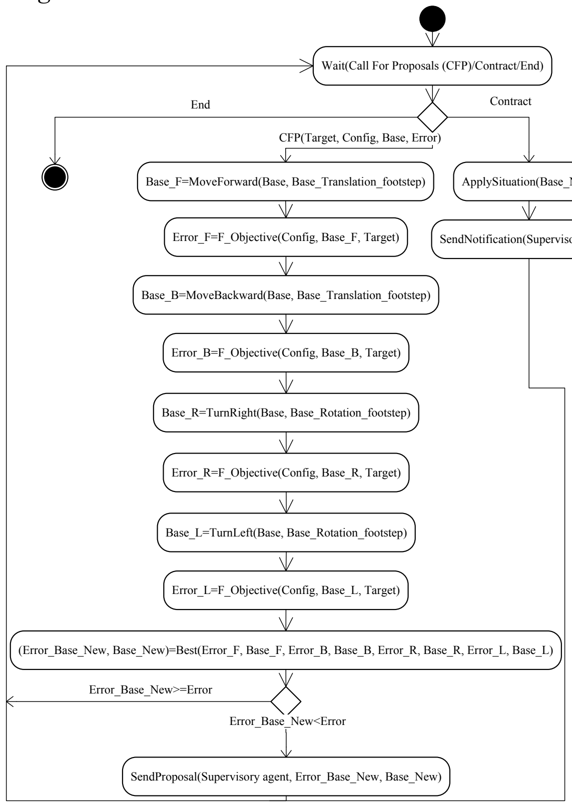

Wait(Call For Proposals (CFP)/Contract/End)
End
Contract
CFP(Target, Config, Base, Error)
Base_F=MoveForward(Base, Base_Translation_footstep)
ApplySituation(Base_New)
Error_F=F_Objective(Config, Base_F, Target)
SendNotification(Supervisory agent)
Base_B=MoveBackward(Base, Base_Translation_footstep)
Error_B=F_Objective(Config, Base_B, Target)
Base_R=TurnRight(Base, Base_Rotation_footstep)
Error_R=F_Objective(Config, Base_R, Target)
Base_L=TurnLeft(Base, Base_Rotation_footstep)
Error_L=F_Objective(Config, Base_L, Target)
(Error_Base_New, Base_New)=Best(Error_F, Base_F, Error_B, Base_B, Error_R, Base_R, Error_L, Base_L)
Error_Base_New>=Error
Error_Base_New<Error
SendProposal(Supervisory agent, Error_Base_New, Base_New)

Fig. 5. Behavior diagram of the *Mobile base agent*

2.3 Supervisory agent

After the introduction of the coordinates of *Effector*$_{Fin}$ by the operator, the *Supervisory agent* verifies if it is reachable. If it is not, the agent terminates the process. Otherwise, the agent computes the initial objective function (ξ_{Init}) corresponding to *Effector*$_{Fin}$. Next, the agent sends ξ_{Init}, *Configuration*$_{Init}$ and *Base*$_{Init}$ to all the other agents. After that, the *Supervisory agent* will wait for their responses (proposals). After receiving all these information, the agents controlling the robot (*Joints* and *Mobile base* agents) will perform, in parallel and independently, their actions to choose the best new configurations with the different errors. Thereafter, the *Supervisory agent* will receive replies sent by the agents of the system, i.e., the best new joint configuration chosen by each *Joint agent*, the new situation of the mobile base chosen by the *Mobile base agent* and their best corresponding errors (*Configuration_I_New*, *ξ_I_New* (i=1… dof), *ξ_Base_New*, *Base_New*). After that, the *Supervisory agent* selects the best response that minimizes the objective function (*ξ_New*). If it is optimal, the *agent* will terminate the process by sending a *task-end message* to the other agents. Otherwise, the *Supervisory agent* sends the chosen values to be applied on the robot. This process continues until reaching the goal (ξ_{Fin} optimal). Fig. 6 illustrates the behavior of the *Supervisory agent* to reach *Effector*$_{Fin}$ by the end-effector of the robot.

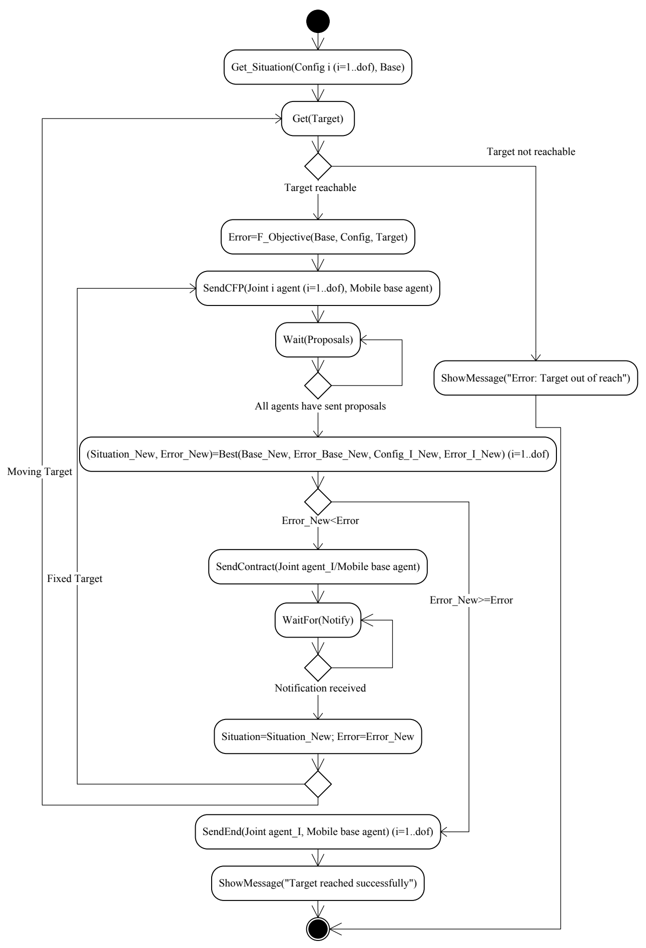

Get_Situation(Config i (i=1..dof), Base)
Get(Target)
Target not reachable
Target reachable
Error=F_Objective(Base, Config, Target)
SendCFP(Joint i agent (i=1..dof), Mobile base agent)
Wait(Proposals)
ShowMessage("Error: Target out of reach")
All agents have sent proposals
(Situation_New, Error_New)=Best(Base_New, Error_Base_New, Config_I_New, Error_I_New) (i=1..dof)
Moving Target
Error_New<Error
SendContract(Joint agent_I/Mobile base agent)
Fixed Target
Error_New>=Error
WaitFor(Notify)
Notification received
Situation=Situation_New; Error=Error_New
SendEnd(Joint agent_I, Mobile base agent) (i=1..dof)
ShowMessage("Target reached successfully")

Fig. 6. Behavior diagram of the *Supervisory agent*

3. SIMULATION SCENARIOS

JADE (*Java Agent DEvelopment Framework*) (Bellifemine et al., 2008) has been used as an implementation tool for the proposed control system. This open source platform provides basic middleware-layer functionalities and simplifies the realization of distributed applications using the software agent abstraction.

It should be noted that all the studies found in the literature tested the performances of the multi-agent control approaches on a simple case of a mobile manipulator in two dimensions. Unfortunately, no works were done in three dimensions.

In this section, we will present some validation scenarios of the proposed control system by using the *RobuTER/ULM*. The different parameters of this mobile manipulator can be found in (Hentout et al., 2010). The distances are given in millimeters (mm) and the angles in degrees (°).

3.1 Validation scenarios

The validation tasks consist of reaching a final situation given by $Effector_{Fin}$. Tab. 1 illustrates the initial values for all the tasks considered in this work.

Table 1. Initial conditions of the considered tasks

Task	Effector$_{Fin}$	Base$_{Init}$	Configuration$_{Init}$	ξ_{Init} (mm)
01	(-330, -630, 1080)	(0, 0, 0)	(0, 0, 0, 0, 0, 0)	1126.9129
02	(-4260, 0, 665)	(0, 0, 0)	(0, 0, 0, 0, 0, 0)	4698.9355
03	(-2408, -108, 1472)	(0, 0, 0)	(0, 60, 0, 0, 32, 0)	3114.8048
04	(-2400, -63, 1325)	(0, 0, 0)	(0, 87, 0, 0, 5, 0)	2946.8779
05	(-2400, -67, 1320)	(0, 0, 0)	(0, 87, 0, 0, 5, 0)	2946.8226

3.2 Pseudo-classical approach

In order to provide a comparison support, the previous tasks have been carried out previously in (Hentout et al., 2010) by using a classical approach (based on *IKM*). The main results are presented in Tab. 2:

Table 2. Results of the pseudo-classical approach

Task	Base$_{Fin}$	Configuration$_{Fin}$	ξ_{Fin} (mm)
01	(0, 0, 0°)	(60, 61, 30, 95, -15, 0)	4.8689
02	(-3440, 13, 12°)	(20, 32, 28, 0, 0, 0)	5.4928
03	(-1920, 2, 15°)	(37, 52, 61, 73, -52, 28)	6.67
04	(-1670, 0, 0°)	(5, 49, 63, -13, -22, -78)	12.3714
05	(-1560, 0, 0°)	(5, 44, 69, -12, -23, -79)	33.762

3.3 Multi-agent approach

The selected simulation footsteps (*Joint footstep*, *Base Translation footstep* and *Base Rotation footstep*) for the execution of the different tasks by the robot are given in Tab. 3. The determination of such footsteps will be the object of another work.

Table 3. Simulation parameters for the tasks execution

Agent	Action	Footstep
Joint agent	MoveUp	Joint footstep = 1°
	MoveDown	
Mobile base agent	MoveForward	Base Translation footstep = 5mm
	MoveBackward	
	TurnRight	Base Rotation footstep = 1°
	TurnLeft	

3.3.1 Scenarios without breakdown

For these first scenarios, we consider that the mobile base and all the articulations of the manipulator are functional. The obtained results are shown in Tab. 4:

Table 4. Obtained results without breakdown

Task	Base$_{Fin}$	Configuration$_{Fin}$	ξ_{IFin} (mm)	Iterations
01	(0, 0, -2°)	(6, 58, 8, 0, -1, 0)	0.7374	75
02	(-3455.07, -492.50, -3)	(-34, -5, 62, 0, -3, 0)	1.3767	810
03	(-1707.73, -243.43, -3)	(-10, 60, 32, 4, 32, 0)	1.4549	395
04	(-1427.56, -203.24, -3)	(-10, 9, 100, 12, 6, 0)	1.4203	1647
05	(-1811.68, -258.24, -3)	(-17, 78, 8, -6, 4, 0)	1.1338	411

3.3.2 Scenarios with breakdowns of some joints

Now, we show how the system reacts in fault cases. The proposed multi-agent control approach is designed to be fault-tolerant. We suppose that the breakdown of the joint 3 and 4 appears at time t=0. The breakdowns are at $q_3=q_{3Init}=0$ and $q_4=q_{4Init}=0$. The obtained results are given in Tab. 5:

Table 5. Obtained results with breakdowns of some joints

Task	Base$_{Fin}$	Configuration$_{Fin}$	ξ_{Fin} (mm)	Iterations
01	(-18.45, -11.96, 10°)	(41, 59, **0**, **0**, 17, 0)	0.9664	148
02	(-3880.77, -553.19, -3°)	(-56, 22, **0**, **0**, 25, 0)	15.7449	905
03	(-1925.53, -274.47, -3)	(-16, 87, **0**, **0**, 40, 0)	22.6522	444
04	(-1836.43, -261.77, -3°)	(-18, 79, **0**, **0**, 25, 0)	0.3878	422
05	(-1810.69, -258.10, -3°)	(-17, 73, **0**, **0**, 40, 0)	2.1747	1933

3.3.3 Scenarios with breakdown of the mobile base

Other scenarios are shown to test the reaction of the system in fault cases. We assume that the breakdown of the mobile base occurs at time t=0. The breakdown is at $Base(x_B, y_B, \theta_B)=Base_{Init}(x_B, y_B, \theta_B)_{Init}$=(0, 0, 0). Tab. 6 gives the obtained results:

Table 6. Results with breakdown of the mobile base

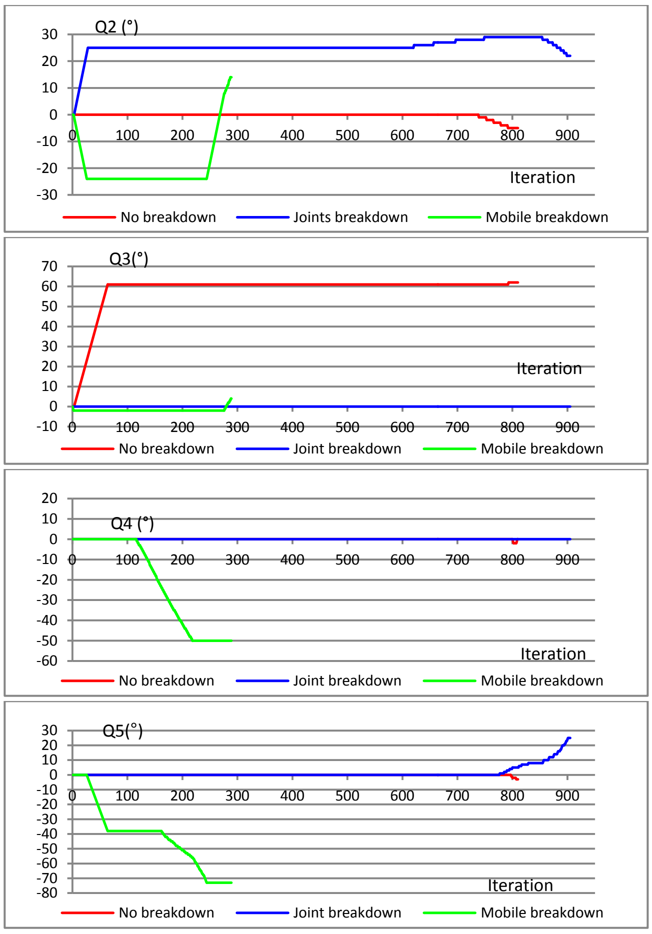
Task	Base$_{Fin}$	Configuration$_{Fin}$	ξ_{Fin} (mm)	Iterations
01	**(0, 0, 0)**	(-95, 53, 11, 88, 40, 0)	54.0627	312
02	**(0, 0, 0)**	(-95, 14, 4, -50, -73, 0)	4004.0195	289
03	**(0, 0, 0)**	(-95, 87, 21, 90,40, 0)	2172.9592	242
04	**(0, 0, 0)**	(-95, 87, 1, 90, 40, 0)	2173.7636	222
05	**(0, 0, 0)**	(-95, 87, 0, 90, 40, 0)	2173.0541	221

The following figures give the variations of the joints, the trajectories of the mobile base and the variations of the positioning errors of the end-effector for the first and the second tasks in all the cases (no breakdown, breakdown of some joints of the manipulator and breakdown of the mobile base):

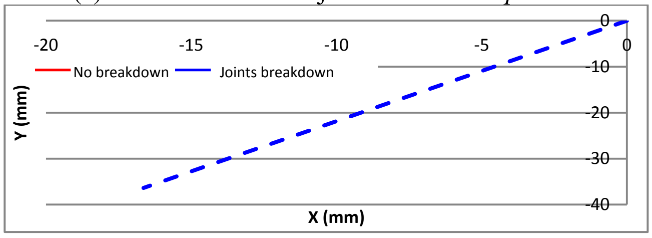
- The red lines are the obtained results in normal case (first scenarios).
- The blue lines represent the results in case of a break-down of some joints (second scenarios).

- The green curves represent the obtained results in case of the breakdown of the mobile base (last scenarios).

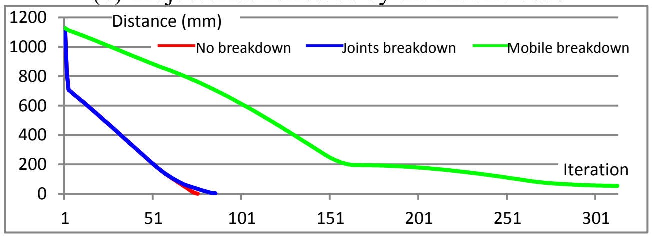
Fig. 8 shows the main results obtained for the first scenario.



(a) Variations of some joints of the *manipulator*

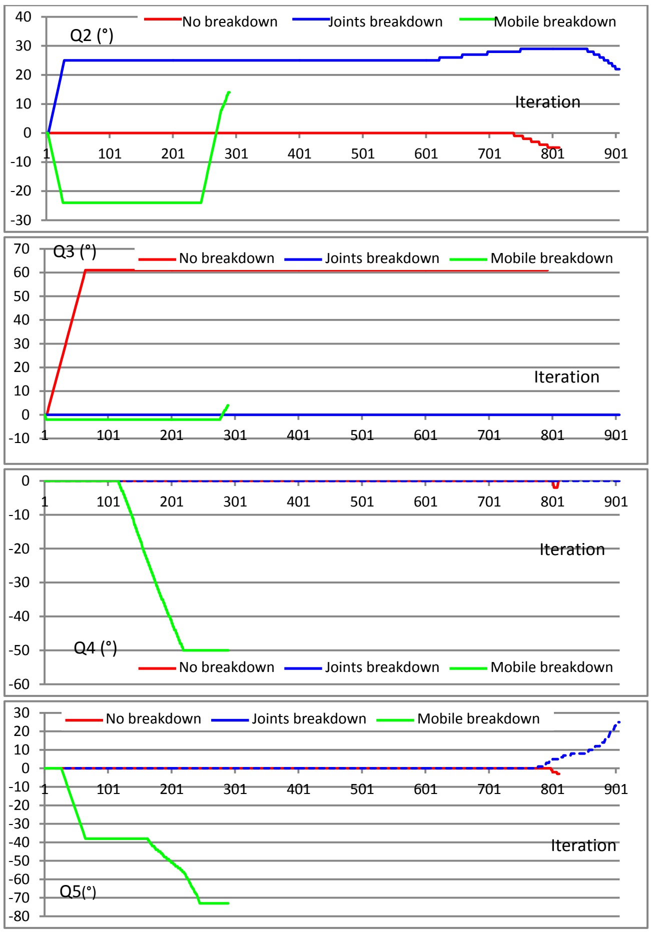


(b) Trajectories followed by the mobile base

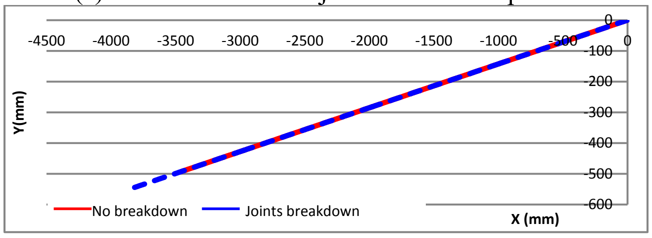


(c) Evolution of the positionning errors of the end-effector
Fig. 8. Main results obtained for the first scenario

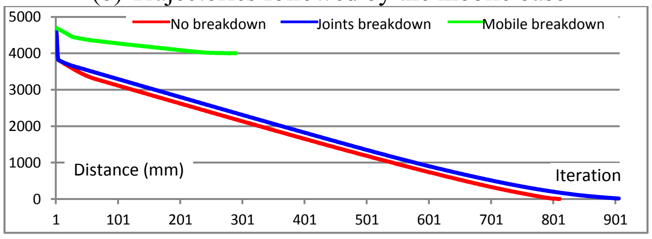
Fig. 9 shows the main results obtained for the second scenario.



(a) Variations of some joints of the manipulator



(b) Trajectories followed by the mobile base



(c) Evolution of the positioning errors
Fig. 9. Results obtained for the second scenario

3.4 Discussion of obtained results

The obtained results using the multi-agent approach were much better comparing those using a pseudo-classical approach. It is important to note that this latter approach wouldn't operate properly if any fault occurs.

For the first scenarios, the robot was able to carry out correctly, and with a good precision, the positioning of its end-effector at *Effector*$_{Fin}$. Concerning the second (resp. the last) scenario, despite the dysfunction of the third and fourth joints of the manipulator (resp. the mobile base), the other agents, still operational controlling the other joints, worked all together to conceal the fault and place the end-effector as close as possible to *Effector*$_{Fin}$ offering, consequently, a minimum service.

4. CONCLUSIONS AND FUTURE WORKS

The presented approach assigns a reactive agent (*Joint agent*) to each dof of the manipulator, a hybrid agent (*Mobile base agent*) for the mobile base, and a hybrid *Supervisory agent* for the coordination of the precedent agents. Each agent has its own local goal, to be reached independently from the other agents, which consists of bringing the end-effector as close as possible to the final situation *Effector*$_{Fin}$. The implementation of the approach used *JADE* which is one of the most interesting multi-agent development frameworks.

The proposed approach was validated via simulation tasks in different cases with *RobuTER/ULM*. The obtained results show that the proposed approach is generic. If the mechanical structure of the robot changes, all we have to do is to change the *DKM* of the robot and associate the required number of agents. In addition, complex mathematical models (*IKM*) don't have to be computed; while offering accuracy similar to classical approaches. Moreover, the proposed approach needs only some geometric formulas and, consequently, requires very little computing power. Finally, the approach is fault-tolerant to failures without adding a specific treatment and, if an agent breaks down, the system provides good result.

The future planned works concern the improvement of the *JADE*-based simulation environment by its integration into the simulator of *RobuTER/ULM* (Akli et al., 2010). This latter will be, next, connected to the real robot accomplishing real tasks. Then, a module of sensors management and obstacles avoidance will be developed and integrated into the *Mobile base agent*. Finally, we will validate the whole of the proposed multi-agent system via more complex scenarios (displacement, object grasping, etc.) to judge its relevance.

REFERENCES

Akli, I., Hentout, A., Bouzouia, B. and Daoud, S. (2010), "Design and Development of Mobile Manipulator Simulator: Application on the *RobuTER/ULM* Mobile Manipulator". *International Conference on Modeling, Simulation and Control (ICMSC2010)*, pp. 370-374, Egypt.

Bayle, B. and Fourquet, J.-Y., Renaud, M., (2003) "From Manipulation to Wheeled Mobile Manipulation: Analogies and Differences", *The International IFAC Symposium on Robot Control (Syroco2003)*, pp. 97-104, Wroclaw, Poland.

Bellifemine, F. L., Caire, G., Poggi, A. and Rimassa, G., (2008), "JADE: A software framework for developing multi-agent applications. Lessons learned", *Information and Software Technology*, 50, pp. 10–21.

Duhaut, D., (1999) "Distributed Algorithm for High Control Robotics Structures". *International Conference on Artificial Intelligence*, Vol. 1, pp. 45-50.

Erden, M. S., Leblebicioglu, K. and Halici, U., (2004) "Multi-agent System-Based Fuzzy Controller Design with Genetic Tuning for a Mobile Manipulator Robot in the Hand Over Task". *Journal of Intelligent and Robotic Systems*, 39(3), pp. 287-306.

Hentout, A., Bouzouia, B., Akli, I. and Toumi, R., (2010), "Mobile Manipulation: A Case Study". *Robot Manipulators, New Achievements, Aleksandar Lazinica and Hiroyuki Kawai (Ed.)*, pp. 145-167.

Hentout, A., Messous, M. A., Oukid, S. and Bouzouia, B., (2013), "Multi-agent Fuzzy-based Control Architecture for Autonomous Mobile Manipulators: Traditional Approaches and Multi-agent Fuzzy-based Approaches". *The International Conference on Intelligent Robotics and Applications (ICIRA2013)*, pp. 679-692, Busan, Korea.

Joukhadar, A., (1997) "Simulation dynamique et applications robotiques", Ph.D. Thesis in Computer science, Polytechnic National Institute of Grenoble.

Medeiros A. A. D., (1998) "A Survey of Control Architectures for Autonomous Mobile Robots". *Journal of the Brazilian Computer Society*, 4(3), Campinas.

Nebot, P., Saintluc, G., Berton, B. and Cervera, E., (2004) "Agent-based Software Integration for a Mobile Manipulator", *The 2004 IEEE International Conference on Systems, Man and Cybernetics (SMC'04)*, pp. 6167-6172, The Hague, The Netherlands.

Nikoobin, A. and Rahimi, H. N., (2009) "Analyzing the Wheeled Mobile Manipulators with Considering the Kinematics and Dynamics of the Wheels", *International Journal of Recent Trends in Engineering, 1(5)*, pp. 90-92.

Wen, J. and Xing, H., (2004) "Multi-agent Based Distributed Control System for an Intelligent Robot", *The International Conference on Services Computing (SCC'04)*, pp. 633-637, China.

Appendix

The parameters used in the paper are given as follows:

- *Effector*$(x_E, y_E, z_E, \psi_E, \theta_E, \varphi_E)$: it is the current situation of the end-effector in the absolute frame. The first three values are the position of the end-effector; the last three represent its orientation angles.
- *Base*(x_B, y_B, θ_B): it is the current situation (position and orientation) of the mobile base in the absolute frame.
- *Configuration*$(q_1, ..., q_{dof})$: it is the current configuration of the end-effector in the manipulator frame.
- *Effector*$_{Init}(x_E, y_E, z_E, \psi_E, \theta_E, \varphi_E)_{Init}$: it is the initial situation of the end-effector.
- *Base*$_{Init}(x_B, y_B, \theta_B)_{Init}$: it represents the initial situation of the mobile base.
- *Configuration*$_{Init}(q_1, ..., q_{dof})_{Init}$: it corresponds to the initial configuration of the end-effector.
- *Effector*$_{Fin}(x_E, y_E, z_E, \psi_E, \theta_E, \varphi_E)_{Fin}$: it represents the final situation of the end-effector (imposed target).
- *Base*$_{Fin}(x_B, y_B, \theta_B)_{Fin}$: it is the final situation of the mobile base.
- *Configuration*$_{Fin}(q_1, ..., q_{dof})_{Fin}$: it corresponds to the final configuration of the end-effector.