A cooperative multi-agent robotics system: Design and modelling

Por: Cena, CG (Garcia Cena, Cecilia)[1,3]; Cardenas, PF (Cardenas, Pedro F.)[2]; Saltaren-Pazmino, R (Saltaren-Pazmino, Roque)[1]; Puglisi, L (Puglisi, Lisandro)[1]; Santonja, RA (Aracil Santonja, Rafael)[1]

EXPERT SYSTEMS WITH APPLICATIONS
Volumen: 40

Número: 12

Páginas: 4737-4748
DOI: 10.1016/j.eswa.2013.01.048
Fecha de publicación: SEP 15 2013
Ver información de revista

Resumen

This paper presents the development of the robotic multi-agent system SMART. In this system, the agent concept is applied to both hardware and software entities. Hardware agents are robots, with three and four legs, and an IP-camera that takes images of the scene where the cooperative task is carried out. Hardware agents strongly cooperate with software agents. These latter agents can be classified into image processing, communications, task management and decision making, planning and trajectory generation agents. To model, control and evaluate the performance of cooperative tasks among agents, a kind of Petri Net, called Work-Flow Petri Net, is used. Experimental results shows the good performance of the system.
(C) 2013 Elsevier Ltd. All rights reserved.

Palabras clave

Palabras clave de autor: Robotic agent; Mechatronics; Mobile robot kinematics; Mobile robot motion-planning; Cooperative systems

Información del autor

Dirección para petición de copias: Cena, CG (autor para petición de copias)
Escuela Univ Ingn Tenc Ind, Ronda Valencia 3, Madrid 28012, Spain.

Direcciones:

[1] Univ Politecn Madrid, Ctr Automat & Robot, Madrid 28006, Spain

[2] Univ Nacl Colombia, Bogota, Colombia

[3] Escuela Univ Ingn Tenc Ind, Madrid 28012, Spain

Direcciones de correo electrónico:cecilia.garcia@upm.es

Financiación

Entidad financiadora	Número de concesión
Spanish Government CICYT Project	DPI2009-08778
Comunidad de Madrid	P2009/DPI-1559

Ver texto de financiación

Editorial

PERGAMON-ELSEVIER SCIENCE LTD, THE BOULEVARD, LANGFORD LANE, KIDLINGTON, OXFORD OX5 1GB, ENGLAND

Categorías / Clasificación

Áreas de investigación: Computer Science; Engineering; Operations Research & Management Science

Categorías de Web of Science: Computer Science, Artificial Intelligence; Engineering, Electrical & Electronic; Operations Research & Management Science

Información del documento

Tipo de documento: Article

Idioma: English

Número de acceso: WOS:000319239700001

ISSN: 0957-4174

Información de la revista

- **Impact Factor:** Journal Citation Reports®

Otra información

Número IDS: 148JC

Referencias citadas en la Colección principal de Web of Science: 32

Veces citado en la Colección principal de Web of Science: 1

A cooperative multi-agent robotics system: Design and modelling

Cecilia Garcia Cena Pedro F. Cardenas Roque Saltaren Pazmino Lisandro Puglisi
Rafael Aracil Santonja

A B S T R A C T

This paper presents the development of the robotic multi-agent system SMART. In this system, the agent concept is applied to both hardware and software entities. Hardware agents are robots, with three and four legs, and an IP-camera that takes images of the scene where the cooperative task is carried out. Hardware agents strongly cooperate with software agents. These latter agents can be classified into image processing, communications, task management and decision making, planning and trajectory generation agents. To model, control and evaluate the performance of cooperative tasks among agents, a kind of Petri Net, called Work-Flow Petri Net, is used. Experimental results shows the good performance of the system.

1. Introduction

The field of multi-agent robotic systems (MARS) arose during the late 80s when several researchers started to work with mobile robots performing coordinated task (Arai et al., 1989; Asama et al., 1989; Beni, 1988). Since this early research, the scientific interest in multi-agent robotic systems (MARS) has increased considerably, (Alami et al., 1997; Balch and Arkin, 1998; Deloach et al., 2002; Fiorino and Tessier, 1998; Kok et al., 2003, 2005).

The increasing interest in MARS is due to a number of real world problems that are best modelled using a set of agents instead of a single agent. A multi-agent system is composed of a number of agents that are able to interact with each other in order to complete a set of goals. Therefore, MARS topics involves all the disciplines in robotics field like distributed control (Wang and Silva, 2010), perception (Franchi et al., 2009; LaValle, 2011; Liu et al., 2010), cognition (Bibel, 2010; Mehrjerdi et al., 2010), coordination (Kitts and Mas, 2009) and configuration (Wang and de Silva, 2010) among others.

However, multi-robot systems bring new problems that must be solved in order to increase the success of this "new" area in robotics: cooperation and coordination for being a social team. In Chen et al. (2010), a system of cooperative work where the robot responds to the targeted task and to temporary restrictions in a framework of individual automatic decision control and communications strategy through specifications of global task, is developed. The formulation of the problem is discrete. The Khepera robots are modelled as agents and the environment as a discrete graph. The execution of the task could be carried out individually or collectively.

Path planning in multi-robot Systems is another important field of research. In concept, considering a group of robots being able to carry out a task in an autonomous manner in changing environments, mainly depends on the sensorization and multi-planning of trajectories, which can be defined as trajectory planning in real time and in parallel with dynamic environment. In Du et al. (2011), a novel procedure is presented for the recompilation of future information and its quality in the planning process. The approach to the solution is associated to the problem in the stochastic dynamic programming. In Kolling and Carpin (2010), evasion-persecution technique in trees or graphs is used. The robotic agents perform a search and evasion on the graph, which models detecting intruders in the complex environments. In Minguez and Montano (2009), an evasion method is proposed (in the detection layer of collision), which considers the exact geometry, the kinematics and the dynamic effects of the mobile robot. The idea is to project distance measurements in a space where the robot can be regarded as a holonomic point. In Peasgood et al. (2008), an alternative method of multi-phase planning that solves the problems of coordinated planning, is discussed. In this method is necessary to make a graph of the environment, based on that, a tree branch is selected. Both are considered as the general free collision route. Besides the tree branch, heuristic approach for its selection is used. It maximizes the number of branches and their distances.

In this paper, the mechatronics development of SMART multi-agent system is presented.

In SMART multi-agent robotic systems, an agent is defined as a software or hardware entity with specific skills and capabilities in order to cooperate and complete a common task. Hardware agents considered in SMART system are:

- IP-Camera Agent. This sensor allows taking pictures of the robotic scene in which robotic agents perform the task in spite of the obstacles.
- A walker robot with four legs and 13 degree of freedom.
- A robot with three legs and eight degree of freedom.

Software agents are developed under C++ language and are listed bellow and :

- Image processing agent: functions developed in order to process the pictures taken by the IP-Camera and obtain information about the position of robots and obstacles in the scene.
- Path planner agent: specific functions that generate and plan the robots's trajectories.
- Communication agent: software that provides communication (TCP/IP and bluetooth) among agents and manages the information packets generated by each agent.
- Decision maker: this agent, called *engine*, has the knowledged of the system and the goals. Therefore it takes decisions about the next step in the system.

A key aspect in the SMART systems is the problem of coordination: the process that ensures that the individual decision of the agents turns out to be a "good" decision for the entire group. In order to ensure the coordination and cooperation between agents, a Petri Net (WF-PN) Work-Flow is used. Fig. 1 summarizes the philosophy of SMART systems. Robotic agents are deployed in a bounded but not structured workspace. Agent camera takes a picture at 15 FPS in bmp format and it is then sent to the central PC via TCP/IP communication. The picture is processed and the information about global position of the robot and obstacles is taken by engine software to plan the task, decide about the next step, which is to re-plan the trajectory (in the agent path-planner) or send a command to the robot by bluetooth communication.

This work is organized as follows, Section 2 describes the electromechanical design of robotic agents. The kinematic model, under screw theory, and the walking s pattern of robots are presented in Section 3. The path planner used for robotics agents is described in Section 4 and experimental results are showed in Section 5. Conclusion and futures works are explained at the end of the article.

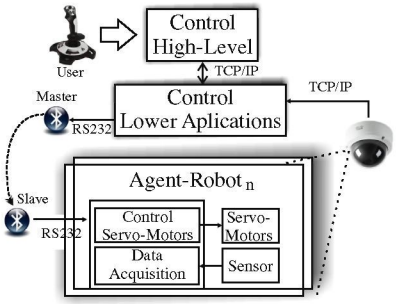

Fig. 1. Multi-agent systems architecture mechatronic.

2. Mechatronics design of the smart system

In this section all the agents developed in this project are described. Firstly, the hardware agent are presented and then the software agent. The mechanical design of robotics agents tries to weigh out complexity of design and performance. Therefore the main goal is to develop a robot with the sufficient degrees of freedom so as to carry out a wide variety of movements. The topics considered in order to achieve the best mechanical design are:

1. The kinematic control must be as simple as possible.
2. The robotic agent must be autonomous.
3. The robotic agent must be able to perform many different task

Two kinds of robotics agents were developed, with three and four legs. Each agent has different locomotion capabilities. The agent with three legs is faster than the agent with four legs, however this latter is able to climb stairs or jump obstacles.

2.1. Agent Robot3L

Fig. 2(a) The module consists of 3 legs and a platform base. The front legs have three degrees of freedom: two rotational and one translational while the hind leg has two rotational degrees of freedom. Likewise, the agent has the ability to go straight and to make turns. Because of the linear actuators, it is possible to move the gravity center up or down. For reasons of balance, it is absolutely necessary that the three legs are resting on the surface. Therefore, the robot moves sliding. Fig. 3(a) shows a picture of the agent built.

2.2. Agent Robot4L

This robot has four legs (see Fig. 2(b)) with three degree of freedom each one (two rotational and one prismatic). This agent can walk using different synchronized patterns like straight line, turns, move the gravity center. There is an extra degree of freedom in the middle of the robot that allows it to stand on two legs. Because it can stand on three legs, the fourth leg can move freely. It is possible to change speed or movement pattern easily. The geometry of the robots is based on the servo-actuators. It thereby reduces the total weight of the structure. Fig. 3(b) shows a picture of the agent built. The robotics agents are controlled by a simple Pololu control board which support Bluetooth protocol. Table 1 summarized the main components used in both prototypes.

2.3. Communication agent

In this project, two different protocols have been used: TCP/IP and Bluetooth. Communications among IP-camera, PCs and software agents have been done using TCP/IP protocol. The communication between robotics agents and control PC has been done on Bluetooth protocol. The central PC is equipped with a Belkin Bluetooth USB adapter. Belkin F8T009 Bluetooth adapter supports Bluetooth version 1.2. SMART robots are equipped with OEMSPA 312iadapter by ConnectBlue. Once connected to its host system and configured, the Serial Port Adapter can communicate, using Bluetooth, with a wide range of other Bluetooth enabled devices such as other Serial Port Adapters, mobile phones, handheld computers and laptops.

For the communication with the serial port adapter, baud rate has been changed to 9600, while Pololu mini servocontroller used to control the SMART robot servos, only works at 2400 or 9600 baud. This can be done, using the OEMSPA312i Serial Port Adapter in AT mode. Communications between IP-Camera and control PC and between software agents and control PC is done on TCP/IP

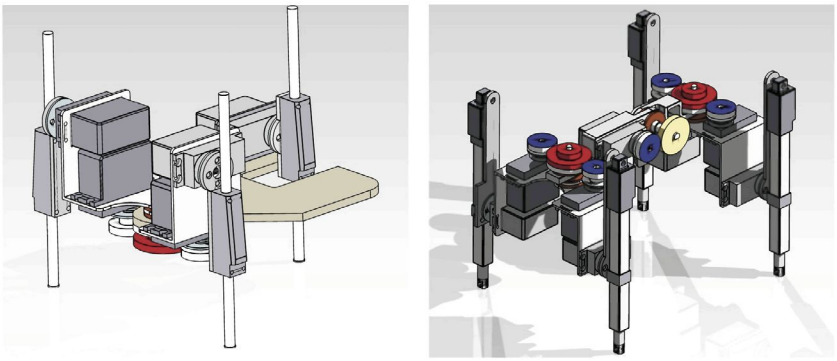

Fig. 2. CAD representation of the robotic modules. (a) Three legs Module (Robot3L), (b) four legs Module (Robot4L).

Fig. 3. Photograph of the robotic modules. (a) Robot3L photograph, (b) Robot4L photograph.

Table 1
Hardware components.

Component	Reference	Qty	Agent
Rotational servo	HS-475	8/5	Robot4L/Robot3L
Prismatic servo 1	L12-Firgelli	4	Robot4L
Prismatic servo 2	LM1247 Faulhaber	3	Robot3L
Servo controller	Micro-Serial Pololu	1	Robot4L/Robot3L
Bluetooth receptor	OEMSPA312i	1	Robot4L/Robot3L
Battery li-PM	EON28	1	Robot4L/Robot3L
IP Camera	Zaapa ZA-CIPRW	1	Camera

protocol. It allows programming software agents in different PC's and to connect them easily and safely.

Message structure The structure designed for the multi-agent environment supports command communication from client to server using a socket. The messages have a defined structure including the action identifier, some parameters and a character to indicate the end of a command. The protocol designed for this communication can be extended when new commands are included in the system. The action identifier is a single number that is unique for each service the server fulfils. The command associated with a given identifier can take some parameters divided by a comma. While the protocol is a sequence of seven bytes, the % sign is used to divide different commands (Sims et al., 2008, chap. 16). The first byte is the command identifier. This command is related to the task that must be performed. The ID of the SMART robot is given in the second byte, followed by a comma to separate the data. Bytes four and six are the parameters for the command i.e. when the command is for a certain robot to move to a point, the parameters will be an X and Y coordinate, a % sign is used. Socket communication is also used to receive images from the webcam as an input for the computer vision delivered by the controlling software. The webcam acts as a server, sending images over the socket to the kernel reacting on HTTPGET/POST requests from the controlling software. The images are saved as bitmap images with the maximum resolution.

2.4. Image processing agent

The image (see Fig. 4(a)) captured by the IP-Camera is sent by "communication agent" to "process image agent" through sockets. This agent has used some functions of OpenCV, a free library available on the internet. However, specific functions have been programmed in order to obtain the position and orientation of the robotic agents and obstacles. The robotic agents are identified by colors: red, blue, yellow and green. The processing process follows five steps. See photograph processing in Fig. 4:

- A non-linear transformation of the RGB to HSV (Hue, Saturation, Value or Brightness). Because RGB representation is highly sensitive to changes in light intensity, it is convenient to use HSV representation. The angle and center of the tone in each HSV color for each robotic agent are defined. (See Fig. 4(b)). Each pixel has a value that is function of the HSV attributes and then the image is segmented taking in consideration the colors.

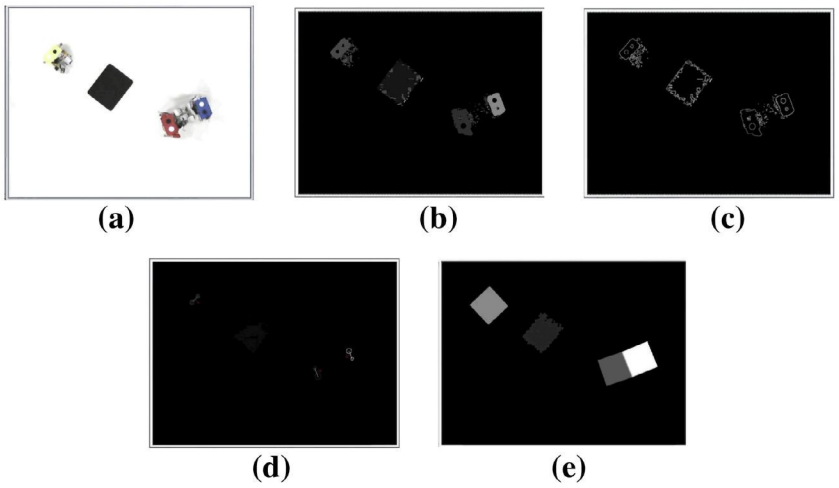

Fig. 4. Image processing. (a) Real imagen captured by IP-camera, (b) non-lineal transformation of the RGB to HSV, (c) get contour of figures, (d) get position of the agent, (e) representation of robot in square model.

1. 0 (white): ground floor.
2. 1 (black): obstacle.
3. 2,3,4,5: robotic agents.

- Filter: The image is subject to a nonlinear mode filter in a region around each pixel (neighborhood 8), eliminating loose pixels in a new image.
- Get the contours: (See Fig. 4(c)). For each pixel, the surrounding region (neighborhood 8) is analyzed and potential changes in pixel value are observed. Outer boundaries are determined by horizontal scanning. Finally, inner contours, robotic agents and obstacles are distinguished using white and black circles arranged in the robotic agents.
- Get position of the robotic agents: the location of each robot in the scene is defined by a point and a slope. Both are obtained by taking the midpoint (geometric center of the robotic agent) of the line that connects the two inner contours defined by black and white circles, and a vector perpendicular to this line. (See Fig. 4(d)).
- Robot Model in the image: each robotic agents is represented as a square with side L, and a equivalent color in a grey scale is assigned. (See Fig. 4(e)).

2.5. Take-decision agent and control architecture

In a multi-agent robotic system, coordination and cooperation among agents are probably the main factor in order to ensure good performance of the task and achievement of the goals. The SMART system is a heterogeneous multi-agent system where the achieved tasks are necessary to be coordinated by hardware and software agents.

Fig. 5 describes the concept of cooperation in SMART system: IP camera takes a picture that has been sent by the communications agent to the image processing agent. The Information about position and orientation of the agent and the obstacles in the scene is shown. This information is again managed by the communication agent and sent, through sockets, to decision-maker agent. The latter agent has information about target to be met by robots. Therefore, it must decide whether to seek the cooperation between robots, in which case the trajectory planner must generate the optimal trajectory for each robot depending on the objective, or simply ask the planner to re-plan the path for the agents as they need. This behavior is modeled by Petri Nets. This theory, proposed by Carl Adam Petri in his Ph.D. thesis in the early 1960s (Petri, 1962), arises as a powerful tool to model concurrent, asynchronous and synchronous systems. In Section 5 dedicated to show experimental results, a model of the system is proposed follow Petri Net concept.

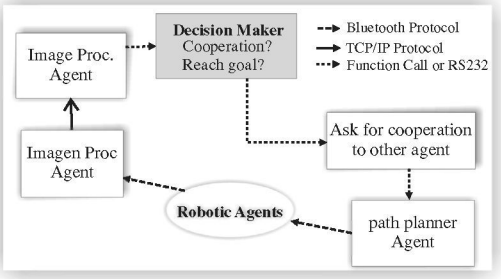

Fig. 5. Scheme of cooperation among agents.

3. Kinematic model and walking patter for robotic agents

One of the most relevant topics of a walking robot is the design of its legs. It must be focused not only on their individual behavior but also on the overall behavior of the robot, considering the desired task and posture that the robot must achieve.

The new SMART robotic walking agents proposes The new SMART robotic walking agents proposes 3 or 4-legged walking robots. Considering 3-legged robots, the posterior leg has 1 DoF, one translational and another rotational. 3 DoF is given to the rest of the legs of the 3-legged robot and the 4-legged robot. They are made up of two rotational joints (named hip and knee), and a prismatic joint for extending the leg, named ext (See Fig. 6). The articulation hip allows the robot to perform changes in direction. The ext articulation allows the robot to increase or decrease the centre of gravity of the robot upon performing same adjustment in its four legs and evading obstacles of smaller or bigger sizes that are present in the trajectory of the extreme part of the leg.

3.1. Direct kinematics

The direct kinematic model of the leg is obtained by applying the successive screw displacement method (Tsai, 1999). This

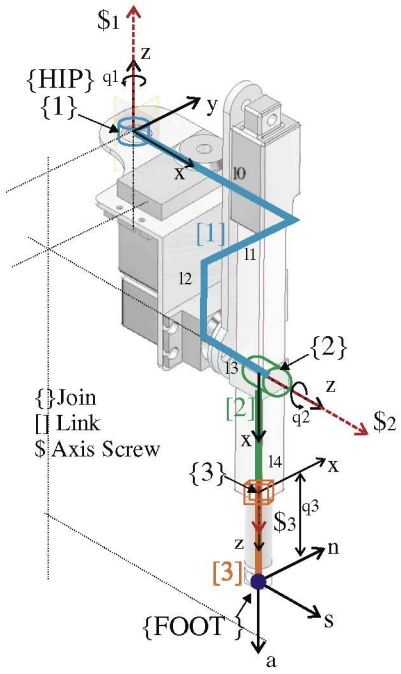

Fig. 6. Leg forward kinematics.

method is based on the identification of the screw axes parameters ($), the reference position (P_o) and the target position (P_{ef}). Let consider the reference position of the mechanism as the one presented in Fig. 6. A fixed frame Oxyz is placed on the hip joint. One screw axis is placed in each joint with the same direction of the joint axis. The targeted position can then be found according to (1).

$$\mathbf{p_{ef}} = A_1A_2A_3\mathbf{p_o} \tag{1}$$

where A_i is the *ith* screw transformation matrix associated to the *ith*-screw axis, and $\mathbf{p_o} = [l_o + l_3, -l_1, -l_2 - l_4, 1]$ are the homogeneous coordinates of the reference position. The parameters of each screw axis are presented in Table 2.

3.2. Inverse kinematics

Given the nature of the kinematic chain of the leg, two possible configurations may result from the inverse kinematic problem. These two possible configurations named: forward knee and backward knee, depends on the selection of the state of q_1.

Consider an arbitrary position of the foot given by $F = [F_x, F_y\ F_z]$, and take a closer look to the projection of the leg over the *XY* plane, as shown in Fig. 7, where *P* is the projection of *F* over the plane *XY*. As it can be seen,

$$d_{AP} = \sqrt{d_{OP}^2 - d_{OA}^2}, \tag{2}$$

$$\alpha = \arctan(d_{AP}, d_{OA}), \tag{3}$$

$$\theta = \arctan(F_y, F_z), \tag{4}$$

where $d_{OP} = \sqrt{F_x^2 + F_y^2}$, and $d_{OA} = l_0 + l_3$.

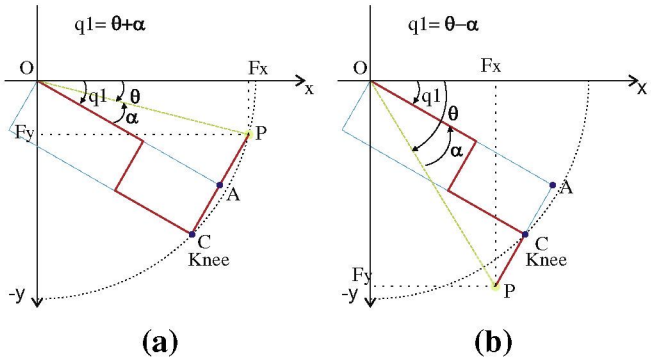

Fig. 7. Possible configurations, (a) Backward-knee (b) forward-knee.

Thus, the state of q_1 for a backward-knee or forward-knee configuration is given by (5)

$$\begin{cases} q_1 = \theta + \alpha, & \textit{Knee front}, \\ q_1 = \theta - \alpha, & \textit{Knee back}. \end{cases} \tag{5}$$

However, there still exists a blind gap where the selection of configuration is uncertain.

With q_1 known, the position of the knee joint can be found according to (6).

$$B = [\cos q_1 d_{OC}, \sin q_1 d_{OC}, -l_2], \tag{6}$$

where $d_{OC} = \sqrt{(l_0 + l_3)^2 + l_1^2}$. The distance from the knee joint up to the foot, defines the state of q_3, as expressed below.

$$q_3 = \|\overrightarrow{BF}\| - l_4, \tag{7}$$

where $\overrightarrow{BF} = \overrightarrow{OF} - \overrightarrow{OB}$.

The state of q_2 is defined as follows,

$$\begin{cases} q_2 = \arccos \dfrac{\overrightarrow{BF}}{\|\overrightarrow{BF}\|}\hat{z}, & \textit{Backward} - \textit{Knee}, \\ q_2 = -\arccos \dfrac{\overrightarrow{BF}}{\|\overrightarrow{BF}\|}\hat{z}, & \textit{Forward} - \textit{Knee}. \end{cases} \tag{8}$$

where $\hat{z} = [0,\ 0,\ -1]$.

3.3. Workspace of the leg

The workspace of the kinematic chain is composed by all these possible positions where the foot can reach without exceeding the physical capabilities of the mechanism. Therefore, several positions for the foot ($F = [F_x, F_y, F_z,]$) are proposed, and using the inverse kinematic model of the kinematic chain, it is verified if the state of the joints resides in the range of work. If the proposed position passes the verification procedure, then the position belongs to the workspace of the kinematic chain, otherwise, it is discarded. For more detail see (Cardenas, 2010). (See Table 3.)

3.4. Instantaneous kinematics

In order to make a synchronized movement of the leg along a desired path with a prescribed speed, the motion of the individual

Table 2
Screw axis parameters.

Joint	s_i	θ_i	s_{0i}	d_i
1	(0,0,1)	q_1	(0,0,0)	0
2	(1,0,0)	q_2	$(0, -l_1, -l_2)$	0
3	(0,0,−1)	0	$(l_0 + l_3, -l_1, 0)$	q_3

Table 3
Amplitude of work for each joint.

Joint	Min	Max	Unity
q_1	$-\pi/3$	$\pi/3$	rad
q_2	$-\pi/3$	$\pi/3$	rad
q_3	0	100	mm

joint must be carefully coordinated. This coordination is achieved relating the joint velocity space and the foot velocity space (end effector velocity space).

According to Tsai (1999), the first-order instantaneous kinematics of a serial robot can be written as (9).

$$\$_n = \sum_{i=1}^{n} \dot{q}_i \hat{\$}_i, \tag{9}$$

where $\$_n = [\omega_n, v_0]^T$ is the resultant twist that describes the infinitesimal displacement of the end effector, $\hat{\$}_i$ is the unit twist associated to the *ith* joint and $\dot{q}_i$ is the intensity of the *ith* twist.

Expressing (9) in a matrix form, and defining $\mathbf{J} = [\hat{\$_1}, \hat{\$_2}, \ldots, \hat{\$_n}]$, the instantaneous kinematic equation can be written as,

$$\dot{\mathbf{x}} = \begin{bmatrix} \omega_n \\ v_0 \end{bmatrix} = \mathbf{J} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix}. \tag{10}$$

Therefore, the columns of J (from now, the jacobian matrix of the kinematic chain), corresponds to the twists associated to each joint.

Considering that the general expressions of an unit twist for a revolute joint and a prismatic joint is given by (11) respectively,

$$\hat{\$}_0 = \begin{bmatrix} \mathbf{s}_n \\ \mathbf{s}_o \times \mathbf{s} \end{bmatrix}, \qquad \hat{\$}_\infty = \begin{bmatrix} \mathbf{0} \\ \mathbf{s} \end{bmatrix} \tag{11}$$

where $\mathbf{s}$ and $\mathbf{s}_o$are given in Table 2, the Jacobian matrix of the leg can be expressed as follows, where $s = \sin()$, $c = \cos()$.

$$J_0 = \begin{bmatrix} \omega_f \\ v_0 \end{bmatrix} = \begin{bmatrix} 0 & c(q_1) & 0 \\ 0 & s(q_1) & 0 \\ 1 & 0 & 0 \\ 0 & l_2 s(q_1) & -s(q_1)s(q_2) \\ 0 & -l_2 c(q_1) & c(q_1)s(q_2) \\ 0 & l_1 & -c(q_2) \end{bmatrix}. \tag{12}$$

However, v_0 is the linear velocity of a point p_0 in the end effector that is instantaneously coincident with the origin of a reference frame in which the twists are expressed (Tsai, 1999). Therefore, the velocity at any point ${}^0p_f = [p_x, p_y, p_z]$ will be given according to the following expressions, (14).

$$\begin{bmatrix} w_f \\ v_f \end{bmatrix} = \begin{cases} {}^0\omega_f = {}^0\omega_f \\ {}^0\mathbf{v}_f = {}^0\mathbf{v}_0 + {}^j\omega_n \times {}^0\mathbf{p}_f \end{cases} \tag{13}$$

Taking into consideration (13), the jacobian matrix can be rearranged, and expressed as,

$$J_f = \begin{bmatrix} 0 & c(q_1) & 0 \\ 0 & c(q_1) & 0 \\ 1 & 0 & 0 \\ p_y & s(q_1)(-p_z + l_2) & -s(q_1)s(q_2) \\ -p_x & (p_z - l_2)c(q_1) & c(q_1)s(q_2) \\ 0 & p_x s(q_1) + l_1 - p_y c(q_1) & -c(q_2) \end{bmatrix}. \tag{14}$$

And the instantaneous kinematic equations can be rewritten as,

$$\begin{bmatrix} {}^0\omega_f \\ {}^0\mathbf{v}_f \end{bmatrix} = J_f \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix}. \tag{15}$$

3.5. Walking pattern for each leg

Fig. 8 shows the robot's movements while it is performing a walking cycle (Estremera, 2003). Additionally, it can be observed the initial position of the robot agent and the description of the phase in every leg, see Fig. 8(a). The gait is divided in two phases; transference phase and support phase. During the transference phase, the leg is not in contact with the floor. The transfer phase is represented by the initial point □ and the final point ■ (see in 8(c)). Both points are linked by a straight line. The arrow shows the direction of the movement. In Fig. 8(c) the black circles indicate the support points of the robot. In order to guarantee the stability of the robot, it is necessary for the center of gravity to be inside the area formed by black circles. In order for the robot to perform a movement in straight line, the direction of the trajectory of the leg has to be parallel with the body as shown in Fig. 8(c). Considering rotation, the direction of the trajectory is tangent to the curve of direction of the rotation. The distance of the step is adjusted so as to guarantee the stability of the robot as shown in Fig. 8(b).

In order to study the timing of the leg's trajectory, it is necessary to define the path to be followed by the foot as function of time. The movement of the foot should be done smoothly and continuously, at least second-order differentiable (Kolter et al., 2008; Wang et al., 2007). Smooth movement of the foot implies that the movement of the robot's trunk is also smooth. In order to ensure the smooth foot trajectory, it is required to observe spatial and temporal restrictions imposed on the kickstand. Some criteria for selecting the path are listed below.

- The orientation of the curve must be normal to the ground during the upward or downward movement of the supporting foot.
- Second derivative of the curve should be continuous.

The walking mode is solved by fixing the paths that make the robot's feet (Goodwine and Burdick, 1997). Using the same path for all the legs and setting an identical relative gap. In the Smart system, it is proposed that legs follow an elliptical path. This path fulfills both restrictions imposed on the motion curve. Fig. 9 shows this trajectory where the middle point is limited by the maximum and minimum length in the third actuator. In order to complete the leg's movement, it is necessary for the leg to come back to the initial position after a relative gap. Since the leg is in contact with the ground, it is assumed that the path described by the leg is a straight line, which should join the starting and ending points (pini and pend) of the gap. See Fig. 9.

4. Path planning

The main goal on the cooperative tasks is that all the robotics agents could reach their target by avoiding obstacles without colliding between them and acting simultaneously.

In this work, an uncouple strategy is used for path planning where the path planning for each robotic agent is treated individually.

The path planning procedure is composed by six main steps, as shown in Fig. 10.

Steps **1** and **2** consist of image capture and image processing of the scene respectively, as previously described in Section 2.4. These steps provide the scene of the workspace as a 640 × 480 grey scale image.

In step **3**, the grey scale image is converted to a binary map (0 means free and 1 occupied) customized for each walking robot (WR), where the rest of the WR are treated as obstacles. Taking the binary map, the C-Space for each WR is obtained, considering that:

1. Robotics agents are ideal mobile robots, i.e. they can perform unconstrained movements in the plane. Therefore, their C-space has three dimension: $[x, y, \alpha]$.

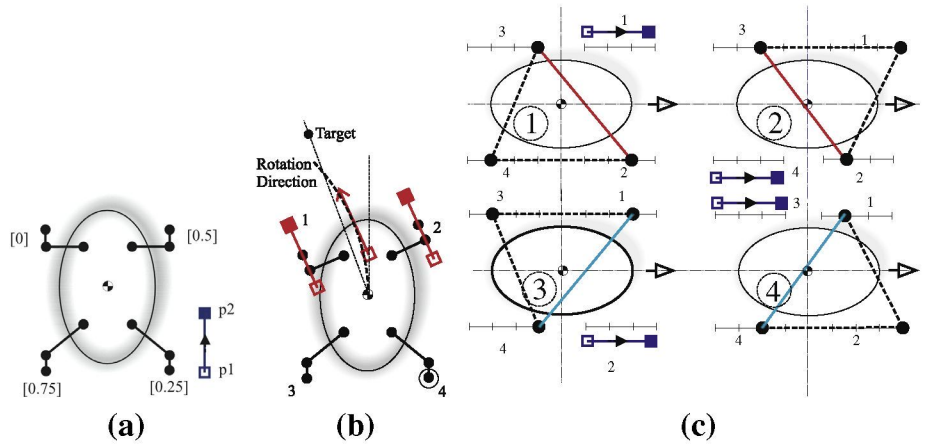

Fig. 8. Crawl gait in straight line and rotation task. (a) Star point-leg and phases-leg to one cycle gait, (b) gait during rotation, (c) leg transition on one gait.

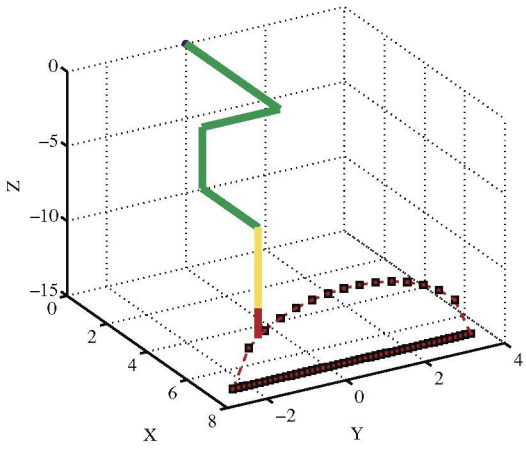

Fig. 9. Trajectory of the leg in the task space.

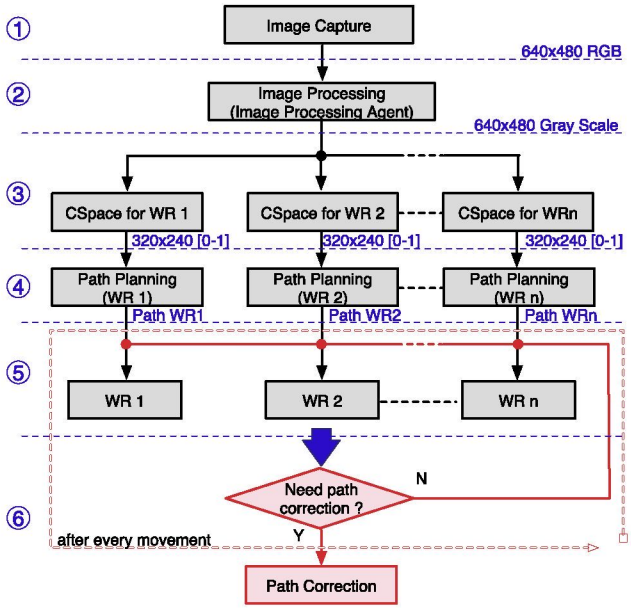

Fig. 10. Schematic diagram of the uncouple path planning strategy.

2. Robotics agents are circles. Therefore, the slices of the C-space are independent of the agent's orientation (α).
3. The workspace is finite, it can be discretized, and it is invariant in time.

Each digitized map is divided into simpler regions, by defining cells as follows:

1. Each cell will have a size of $n \times n$ pixels.
2. The digitized map is explored, and the frontiers of the cells are stablished.
3. Each cell is identified as occupied or free (1 or 0). If one of the cell's pixels is occupied, the entire cell will be defined as "occupied". If all the cell's pixels are free, then the cell will be defined as "free".

In step **4**, the path planning for each WR is performed individually, considering their customized 320×240 binary map.

The planned path is traduced into WR's movements in step **5** (see Section 2). After each completed movement of the WRs, the path tracking and correction algorithm (step **6**) supervises whether the actual paths of the WRs needs to be corrected or not. This algorithm, evaluates if the WRs are out of the preplanned path, and if collision between them could arise.

4.1. Path planning algorithm

The path planning algorithm is executed individually for each robotic agent with a customized map, where the rest of the robotic agents are considered as obstacles.

The algorithm is based on the A^* algorithm, with the followings characteristics:

1. The evaluation function is given by: $f(n) = g(n) + h(n)$, where $h(n)$ is evaluated as the Euclidean norm from the actual node to the target node. The real cost $g(n)$, is evaluated as the quantity of nodes routed from the initial node to the actual node n.
2. Node selection based on priorities. If two nodes have the same cost ($f(n)$), for the next exploration, it will be chosen the node that is toward the direction of the target node.
3. An 8-neighborhood is considered as expansion strategy for the exploration.
4. Auxiliar data lists:
   - **open nodes list**: list of nodes that have already been visited, but not expanded.

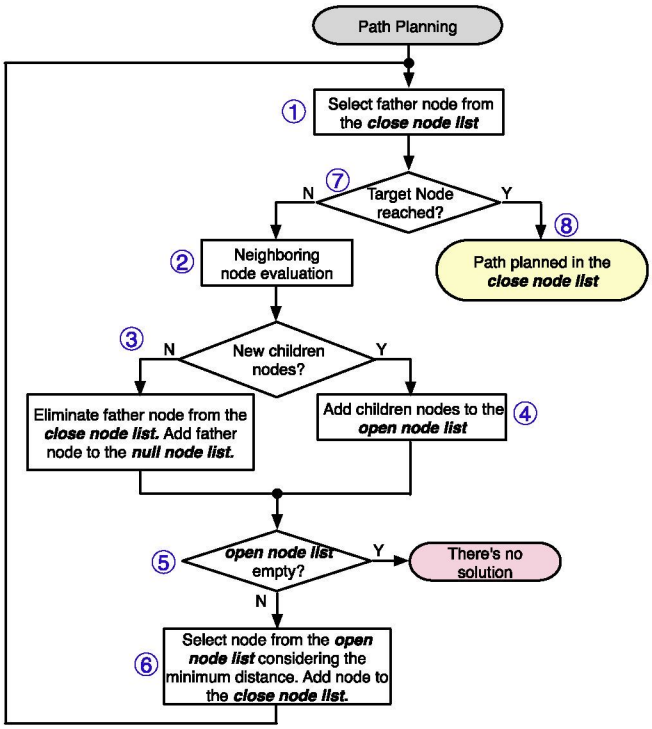

Fig. 11. Schematic flow chart of the path planning algorithm.

- **closed nodes list**: nodes that belong to the path.
- **null nodes list**: nodes with out an exit. Once that they are detected, they are no longer allowed for reevaluation.

5. All nodes are defined by a five field structure. This structure, contains the x and y coordinates of the node, its distance ($g(n)$), its cost ($f(n)$), and its father node.
6. Elimination of unnecessary steps. Those nodes that do not represent a real approximation to the target node, are deleted from the "close node list".
7. After completion of the algorithm, the close node list is routed backwards in order to find the minimum distance for the path.

At the beginning of the path planning algorithm, it verifies that the origin and target point belongs to the free space of the map, and it proceeds as follows (see Fig. 11):

1. The last node inserted in the "close node list" is taken as the father node.
2. The father node is expanded considering an 8 neighborhood.
3. The neighboring nodes will be considered as children nodes if they verify that they were not previously visited and that they belong to the free space. If none of the neighboring nodes verifies it, the father node is eliminated from the "close node list" and it is added to the "null node list".
4. $f(n)$ and $g(n)$ are evaluated for all the new children nodes, and are added to the "open node list". If any of the new children nodes are already listed in the "open node list", it will be kept in the list the element with lower $g(n)$.
5. The "open node list" is verified. If the list is empty, then there is no possible path, and the algorithm ends.
6. The next father node is selected from the "open node list". The selected node must signify an approximation to the target node and guarantee continuity to the path. This is achieved by consulting the "close node list" backwards. The selected node must also verify: $g(n) < d_{min}$. Where d_{min} is a distance that is updated in every iteration with the minimum $g(n)$ found in the previous iteration. In the first iteration, d_{min} is set to the sum of all the cells of the binary map.
7. The selected node is introduced in the "close nodes list", and is taken as the father node in the next iteration.
8. If the father node corresponds with the target node, the algorithm finds a solution. From all the possible paths that could be generated with the nodes in the "close node list", it is chosen the path with the minimum distance.

4.2. Path tracking and correction

The path tracking and correction algorithm is executed after the completion of every move performed by the WRs. This algorithm consists of two main functions:

- **Path correction**: It detects whether a robotic agents is no longer following its preplanned path by evaluating the distance of its current location (d_c) as follows:

1. $d_c < d_{min}$: It is considered that WR continues over the trajectory.
2. $d_{min} \leqslant d_c \leqslant r$: It is considered that WR is found far from the trajectory and has to come back to its place.
3. $r < d_c$: It is considered that WR is found very far from the trajectory. Therefore, WR is detained and a new trajectory planning is done.

Where d_{min} corresponds with the distance covered by the WR in a standard displacement, and r is the radius of the circle in which WR is defined.

In case the trajectory of WR needs to be corrected, the algorithm proceeds in the following manner:

1. It finds the optimal point for the reinsertion of the WR to the predefined trajectory. The distance of all points pertaining to the trajectory is thereby calculated. A search radius is defined given by the maximum displacement of the WR in its next move. Considering all possible trajectory points, the farthest point is chosen and moreover, it represents the approach towards the objective.
2. The current localization of WR and the point of insertion to the trajectory define the new advancement of the robot, and the WR is put in movement.
3. Once the point of reinsertion is reached in the trajectory, WR is oriented in the sense of the trajectory. The pace of advancement of the WR is regulated as it approaches the point of reinsertion.
4. WR keeps the predefined trajectory.

- **Collision detection and avoidance**: After the completion of every move performed by the WRs, an image of the scene is captured an analyzed. The distance between the WRs is measured. If this distance is less than two standard step of the WRs, then the WRs that are involved in the possible collision, are detained and a new path planning is performed for the WR with higher priority. The WRs with lower priority are detained until the WR with higher priority surpasses the critical region. Afterwards, they recover their normal behavior. The priority of each WR is defined with the definition of the cooperative task.

5. Experimental result

This section shows the experimental results obtained in the SMART multi-agent system.

5.1. Path planning algorithm

In order to verify the capabilities of the path-planning algorithm, several scenarios were simulated and tested before the algorithm was implemented in the real system. Fig. 12 shows the path found for a spiral map, labyrinth map and zig-zag map type. In all of these experiments, the path was found in less than 30 ms.

After doing the simulations, an experiment where an agent robot in located in the scene, together with various fixed obstacles, is carried out. The trajectory is planned in order for the robot to reach the final point, which is located in front of the scene. It goes through the obstacles without collision.

Two experiments that allow validating the general performance of the software and hardware agents are shown below. The first experiment is about the performance of algorithm A*, which is subject to the modifications carried out (Section 4) when a robotic agent tries to abandon the planned trajectory. In the second experiment, the possible collision of the robot agents is detected. The decision-maker agent forces the robot to detain. It then redefines its path trajectory, considering the first as a fixed obstacle. (See Fig. 13)

Experiment 1: Reinsertion of a robot agent in the predefined trajectory. Fig. 14 shows the results from the first experiment. It can be observed in Fig. 14(a) how agent 1 begins to leave the preplanned path, and the path tracking control detects it. Fig. 14(b) shows the points where the control recognizes as possible reentry points to the former path, and the selection of the farthest one.

Experiment 2: Avoidance collision between two robotics agents. This experiment allows to test the performance of the collision algorithm. Fig. 15(a) shows two walking robots with their respective planned trajectory. The algorithm detects that the current position of Agent 2 is not safe and interferes or could interfere in the trajectory of Agent 1. Therefore, the algorithm stops Agent 2 and performs a new planning for Agent 1 (Agent 1 has the major priority in the task). See Fig. 15(b).

5.2. Cooperation between robotic agents

In this subsection an example of the many possibilities of cooperative tasks is presented. The reader is refereed to García et al. (2012) and Garcia et al. (2010). The agents move inside a metallic structure with a colored IP-Camera placed 2 m above it. The camera takes the actual pictures into the host computer where the engine software runs and carries out recognition and proper decisions based on computer vision results. One of the main objectives is to move a set of robotic agents in a coordinate way to complete the task. In this experiment, a helping task is performed. Suppose that the quadruped robot fell down and the three-leg robot help him to stand up. In this situation, the three-leg robot is just held the four-leg agent.

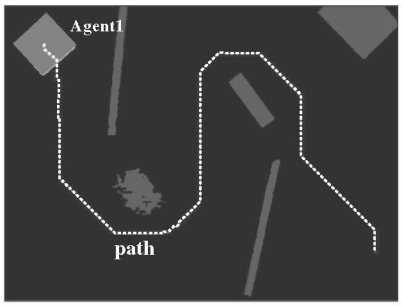

Fig. 13. Path planning of a real robotic agent in a scene with multiple obstacles.

The IP camera used captures images up to 15 fps in VGA format, which is used by the authors. If the processing software of the images is included, the speed is reduced to 11 fps. Nevertheless, the control loop (Fig. 5) is closed with 6 fps, which is good enough considering the speed of the movement of the robot.

The cooperative task is described in Fig. 16. A yellow robotic agent with three legs approach to four-leg robot. The last one fell down and cannot stand up by himself. Therefore, it uses the tree-leg robot for leaning on the three-leg agent. Using the camera information and the kinematic knowledge of the agents, the path planner gives the commands to four-leg robot to stand up.

5.2.1. Modelling of behaviour using Work-Flow Petri Nets

Petri Network (RdP) makes up a theory for modelling systems of flow where events can take place in sequential manner or concurrent. It thereby converts it into a powerful tool for modelling homogeneous or heterogeneous multi-agent systems.

In the SMART system, the RdP is used for modelling the behaviour of every agent and/or a group of agents working for a common objective. Considering the broad variety of Petri nets available, the authors have chosen the Work-Flow kind since it comes with logical transitions that simplify the modelling better than any other conventional network. The practical implementation of the network has been done in C++ with multi-thread architectures as shown in detail in Garcia et al. (2010).

In concept, when cooperation is established between hardware agents as shown in Fig. 16, the robot will move at the same time as long as there is no risk of collision. Upon the detection of a collision, an agent is detained. The other agent considers it as a fixed obstacle. Considering which agents is to detain or continue is determined deductively in accordance to the task being carried out. Formally, a marked RdP is defined as a given tuple in (16):

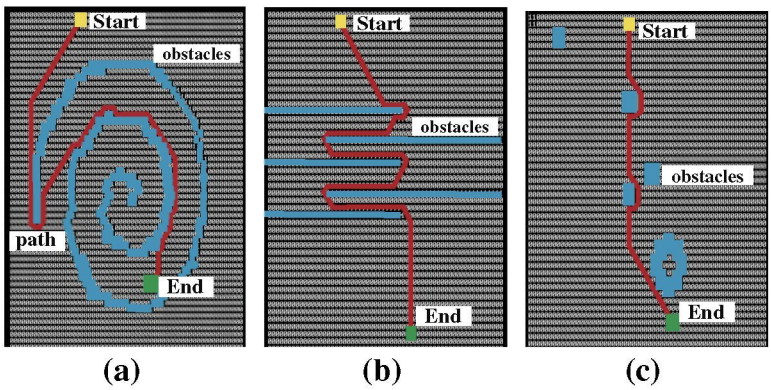

Fig. 12. Path planning experiments results. A 0 corresponds to an empty space and a 1 an occupied space. 12(a): spiral map. 12(b): labyrinth map. 12(c): zig-zag map.

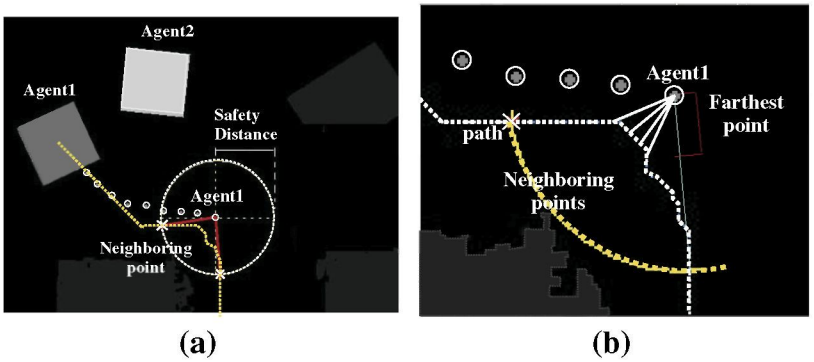

Fig. 14. Recovering preplanned path Section 5.1: a real scenario with a real agent. (a) Agent1's trajectory is out of path, (b) neighboring points for possible reinsertion to former path.

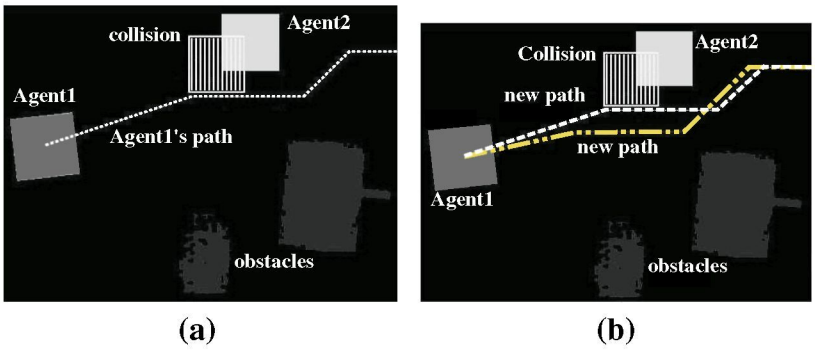

Fig. 15. Collision detection and avoidance experiment. (a) Collision detection (b) new path planning.

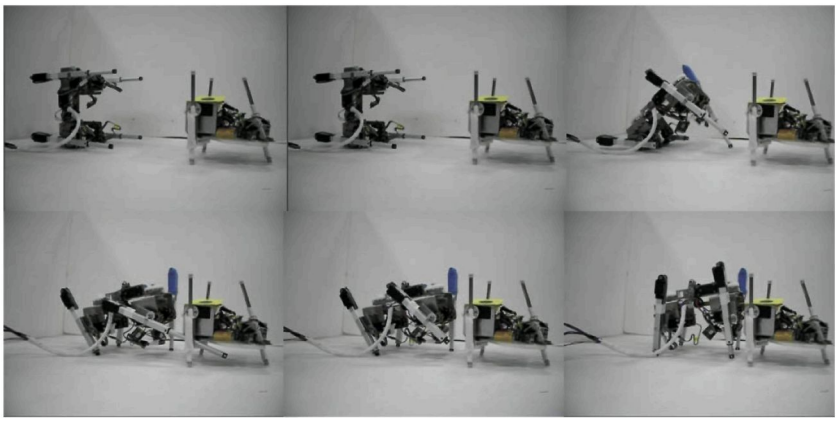

Fig. 16. Example 2: cooperative task between robotics agents.

$$N = (P, T, F, W, M), \tag{16}$$

where P is a finite set of status and T is a finite set of transitions. Both sets verified that $P \bigcap T = \emptyset$ and $P \bigcup T \neq \emptyset$. F is the set of arches such as $F \subset \subset (P \times T) \bigcup (T \times P)$, the arc weight f_{ik} links the place p_i with the transition t_k and it is defined as $w \in W{:}F \rightarrow Na+$ where $Na+ = 1,2,\ldots$ is the set of entire positives. An M mark on network N is a map of $P \rightarrow Na$. $M(p_i)$ denotes the number of marks in $p_i \in P$ place and clearly becomes null. A Petri network marked, $N_M = (N,M)$, has an initial M^0 mark. Fig. 17 shows the model of cooperative behavior developed in this section. This model was performed and simulated in a free distributed software called WoPed (Eckleder and Freytag, 2008). The simulation results are shown in Fig. 18.

In this model, some sub-nets can be distinguished. This WF-PN can be divided into sub-nets in order to model each agent in this way:

- *IP-Camera subnet*, N_M^{IP-C}, models the software of the image capture by the camera placed in the scene. In this net, $P = \{P_1, P_2\}$ is the set of status and defined as P_1: Captured image, P_2: Camera on standby. $T = \{T_1, T_2\}$ and defined as: T_1: Image sent on bmp format to the processing software, T_2: New image captured. In the initial status, $M^0 = [10]^T$ is maintained and $W \in \Re^{4\times 1=I}$. The T_2 transition occurs only in case the camera is not occupied and indicated by the control software (modelled in P_6).

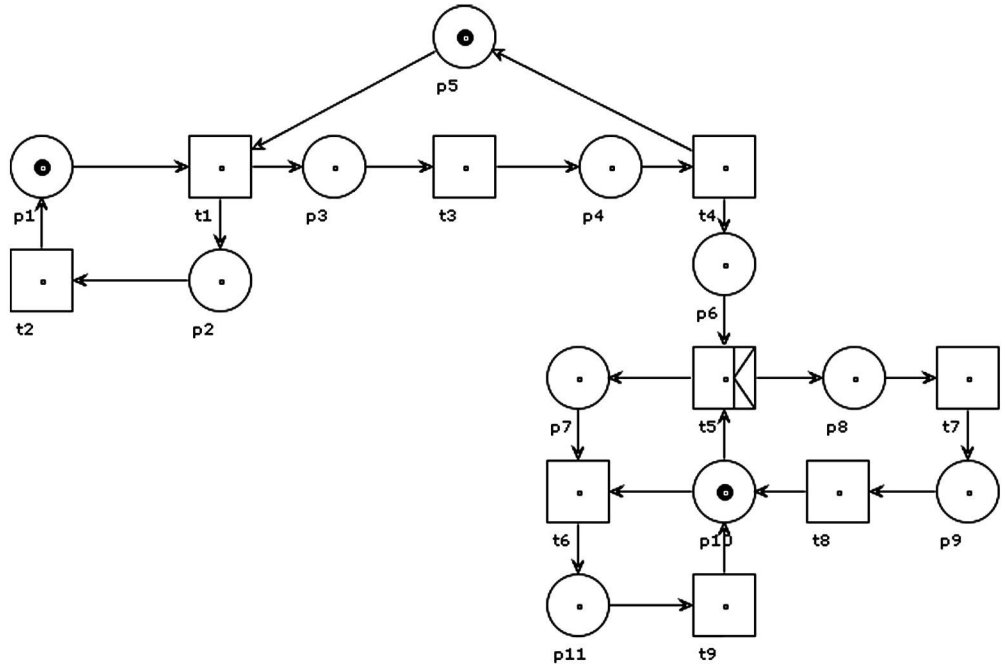

Fig. 17. Model of cooperative task between two smart robotic agents using Petri Nets.

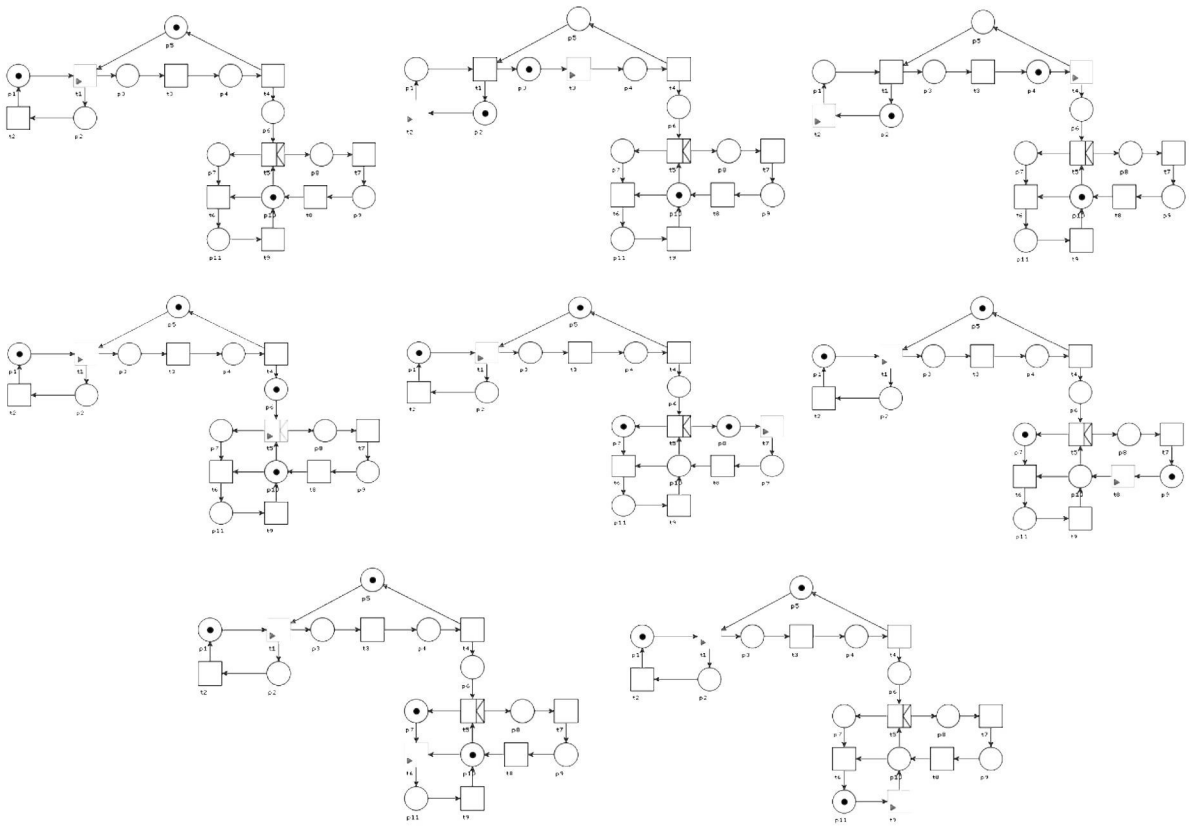

Fig. 18. Model of cooperative task between two smart robotic agents using Petri Nets.

- *Image processing subnet*, N_M^{ID-S}, models the image processing software, in order to get the location of the agents and obstacles. The capture and image processing is done in 11 times per second. Where $P = \{P_3, P_4, P_5\}$ is the set of status and defined as: P_3: Reception of the bmp image through the socket,P_4: identification of agents and obstacles. Extraction of coordinates of position y orientation (as explained in Section 2.4). P_5 on standby for the next image. $T = \{T_3, T_4\}$, where T_3 indicates the

beginning of image processing, T_4 sends the information to the control block of agents. This Communications is done through TCP/IP. In the initial status of the marking $MM^0 = [001]^T$ and $W \in \Re^{5\times 1=I}$.

- *Decision Maker subnet*, N_M^{DM} models the decision maker software agent that is the kernel of the application. Here, $P = \{P_6\}$ that decides what robot agent has to be assisted in order to complete its task. Transition $T = \{T_5\}$ is the AND-SPLIT type that locates a mark in only one status wherein it is connected. The decision taken in P_6 status depends on the information obtained in the image processing and the previous information of the task to perform. In the initial status, $M^0 = [0]$ and $W = \{w_i\} \in \Re/w_i = 1$.
- *Cooperative behavior*, N_M^{CB}. This net models the cooperation between agents. $P = \{P_7, P_8, P_9 P_1 0, P_1 1\}$ is the set of status where P_7 and P_8 will receive a token from AND-SPLIT transition. In P_7 the crashed robot receive an order: "wait for help". While the helper receive the order to start to help (P_8). A trajectory and a final point are sent to helper (T_7, P_9, T_8). When it reaches the final position, a token is placed in P_{10}, therefore the transition T_6 can be fired. After the fire, the crashed robot stars to stand up ($P_1 1$) while the helper stays in the final position (T_9). In this subnet $M^0 = [00000]^T$ and $W \in \Re^{5\times 1=I}$.

The proposal net is simulated using Woped software and the results are shown in the sequence of 18. Here it can be follows the development of the cooperative task.

6. Conclusions

This article presents a modular robotic system called SMART. This system consists of different types of software and hardware agents. In the hardware agents, there are two kinds of robots with three and four legs. In any multi-agent systems, its success largely depends on the communications architecture. Therefore, this article broadly describes the model and protocol used in this system. The software developed to control the system includes all functionality planned for the SMART robots in this first phase of the project. The software can easily be expanded in the future when new tasks will be added to the modular robotic system. The developed software is typical modular robotics architecture. The multi agent system guarantees good performance of cooperation tasks between agent robots; camera; user interface and communications protocol. In the future, an auto-connected architecture will be developed. It will be done by using reciprocal communications. Robots will then be able to help each other. When a single robot cannot finish the task, other robots will help in order to accomplish the task without the interaction of the control software. In the same way, a processor will be added on board in order to allow each module to take decisions by itself.

References

Alami, R., Fleury, S., Herrb, M., Ingrand, F., & Robert, F. (1997). Multirobot cooperation in the martha project. *IEEE Robotics & Automation Magazine*.

Arai, T., Ogata, H., & Suzuki, T. (1989). Collision avoidance among multiple robots using virtual impedance. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems* (pp. 479–485).

Asama, H., Matsumoto, A., & Ishida, Y. (1989). Design of an autonomous and distributed robot system: Actress. *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems* (pp. 283–290).

Balch, T., & Arkin, R. (1998). Behavior-based formation control for multirobot teams. *IEEE Transactions on Robotics and Automation, 12*.

Beni, G. (1988). The concept of cellular robot. In *Proceedings of the third IEEE symposium intelligent control* (pp. 57–61).

Bibel, W. (2010). General aspects of intelligent autonomous systems. In D. K. Pratihar, L. C. Jain (Eds.), *Intelligent autonomous systems* (pp. 5–27).

Cardenas, P. F. (2010). Análisis cinemático de robot modular cuadrupedo. Master thesis, Universidad Politécnica de Madrid, p. 150.

Chen, Haoyao, Sun, Dong, Yang, Jie, & Chen, Jian (2010). Localization for multirobot formations in indoor environment. *IEEE/ASME Transactions on Mechatronics, 15*(4), 561–574.

Deloach, S., Matson, E., & Li, Y. (2002). Applying agent oriented software engineering to cooperative robotics. In *Proceedings of the 15th international florida artificial intelligence research society conference*.

Du, T., Noel, E., & Burdick, J. W. (2011). Robot motion planning in dynamic, uncertain environments. *IEEE Transactions on Robotics*, 1–15.

Eckleder, A., & Freytag, T. (2008). Woped a tool for teaching, analyzing and visualizing workflow nets. *Petri Net Newsletter, 75*, 3–8.

Estremera, J. (2003). Modos de Caminar Libres y Sensores Virtuales para Robots Caminantes. Ph.D. thesis, Universidad Complutense de Madrid.

Fiorino, H., & Tessier, C. (1998). Agent cooperation: A petri net based model. In *Proceeding of international conference on multi agent systems* (Vol. 3, pp. 425–426).

Franchi, A., Freda, L., Oriolo, G., & Vendittelli, M. (2009). The sensor-based random graph method for cooperative robot exploration. *IEEE/ASME Transactions on Mechatronics, 14*(2), 163–175.

García, C., Cárdenas, P. F., Puglisi, L. J., & Saltaren, R. (2012). Design and modeling of the multi-agent robotic system: Smart. *Robotics and Autonomous Systems, 60*(2), 143–153.

Garcia, C., de Clerq, S., SaltarTn, R., Aracil, R., & Verschelde, W. (2010). Development of communication architecture for the multiagent robotic system: Smart. In *Proceedings of the European conference on the use of modern information and communication technologies* (pp. 20–26).

Goodwine, B., Burdick, J. (1997). Trajectory generation for kinematic legged robots. In *IEEE international conference on robotics and automation* (pp. 2689–2696).

Kitts, C. A., & Mas, I. (2009). Cluster space specification and control of mobile multirobot systems. *IEEE/ASME Transactions on Mechatronics, 14*(2), 207–218.

Kok, J. R., Spaan, M., & Vlassis, N. (2003). Multi-robot decision making using coordination graphs. In *Proceedings of the international conference on advanced robotics (ICAR)* (pp. 1124–1129).

Kok, J., Spaan, M., & Vlassis, N. (2005). Non-communicative multi-robot coordination in dynamic environments. *Robotics and Autonomous Systems, 50*, 99–114.

Kolling, A., & Carpin, S. (2010). Pursuit-evasion on trees by robot teams. *IEEE Transactions on Robotics, 26*(1), 32–47.

Kolter, J. Z., Rodgers, M. P., & Ng, A. Y. (2008). A control architecture for quadruped locomotion over rough terrain. In *IEEE international conference on robotics and automation* (pp. 811–818).

LaValle, S. M. (2011). Motion planning. *IEEE Robotics & Automation Magazine, 18*(1), 79–89.

Liu, S., Sun, D., & Zhu, C. (2010). Coordinated motion planning for multiple mobile robots along designed paths with formation requirement. *IEEE ASME Transactions on Mechatronics*, 1–11. http://dx.doi.org/10.1109/TMECH.2010.2070843.

Mehrjerdi, H., Saad, M., & Ghommam, J. (2010). Hierarchical fuzzy cooperative control and path following for a team of mobile robots. *IEEE/ASME Transactions on Mechatronics* (99), 1–11. http://dx.doi.org/10.1109/TMECH.2010.2054101.

Minguez, J., & Montano, L. (2009). Extending collision avoidance methods to consider the vehicle shape, kinematics and dynamics of a mobile robot. *IEEE Transactions on Robotics, 25*(2), 367–381.

Peasgood, M., Clark, C. M., & McPhee, J. (2008). A complete and scalable strategy for coordinating multiple robots within roadmaps. *IEEE Transactions on Robotics, 24*(2), 283–292.

Petri, C. A. (1962). Communication with automata. Ph.D. thesis.

Sims, M., Corkill, D., & Lesse, R. V. ((2008). Automated organization design for multi-agent systems. *Autonomous Agents and Multi-Agent Systems, 16*(2), 151–185.

Tsai, L. W. (1999). *Robot analysis: The mechanics of serial and parallel manipulators*. Wiley-Interscience.

Wang, X., Chen, X., Jia, W., Sun, Y., & Pu, H. (2007). *Forward kinematics analysis and 3-dimmision gait simulation of a MiniQuad walking robot*. IEEE.

Wang, Ying, & de Silva, C. W. (2010). Sequential q-learning with Kalman filtering for multirobot cooperative transportation. *IEEE ASME Transactions on Mechatronics, 15*(2), 261–268.

Wang, Y., & Silva, C. (2010). Sequential q-learning with Kalman filtering for multirobot cooperative transportation. *IEEE/ASME Transactions on Mechatronics, 15*(2), 261–268.