Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

Formal specification and simulation of the robot path planner

M.Yassine Belkhouche
Computer Science and Engineering Department
University of North Texas
Denton, Texas
myb0012@unt.edu

Boumediene Belkhouche
College of Information Technology
UAE University
Al-Ain, UAE
B.Belkhouche@uaeu.ac.ae

***Abstract*—Simulation software plays a very important role in education and research applications. They provide a safe, easy and flexible environment for developing and testing new methods. However developing a simulation software that reflects the exact behavior of the real system is a difficult and complicated task. In this paper we used hybrid process algebra to build a formal model that describes the exact behavior of the path planner module of autonomous mobile robots. The robot path planner is described by a powerful recursive process. The robot moves in four modes heading-regulation, move-to-goal, move-to-inter-goal, and obstacle-avoidance mode. Each motion mode is described by a process. The overall behavior of the path planner is the result of the communication between the four processes. The developed model is then implemented as simulator for mobile robots.**

***Index Terms*—Hybrid Process Algebra, Mobile Robot Simulation, Hybrid Systems, Formal Specification.**

I. Introduction

The real world is a complex, unstructured, and dynamic environment. Autonomous mobile robots which are designed to operate in the real world must have the ability to deal with such a complex environment. A mobile robot is a hardware/software system in which the control software plays a central role. In practice most robot failures are due to a failure in the control software. A software model for the robot control is based on dividing the functions of the system into modules, each supervising a basic activity. This approach decouples the complexity of the control tasks, and facilitates the validation of each module. Well-defined protocols are required to effectively manage the various interactions among the different hardware/software components. To meet the challenge of control design for complex high performance systems and with strict safety requirements, formal hybrid specification methods have been developed. These methods enable developers to rigorously model, test, verify, and simulate such systems. Hybrid automata [1], [2] are powerful modeling languages that support modeling a wide variety of physical phenomena. Hybrid automata integrate diverse models such as differential equations and state machines in single formalism. In this mathematical model the behavior of the system is described by a graph. The continuous state of the system is represented by differential equations, and the discrete state is represented by the graph vertices. The transition between discrete states are represented by edges. Hybrid process algebras [3], [4], [5], [6] are another mathematical theory developed for the specification and analysis of hybrid systems. In this theory the system behavior is expressed in the form of algebraic terms. This theory allows the description and syntax-based analysis of hybrid system in a compositional way. The mathematical expressions for hybrid systems are constructed by means operators, each of which corresponds to a distinct and natural way in which hybrid system can be combined. This theory provides also axioms lifting rules that can be used to establish whether two expressions constructed in different ways represent the same hybrid system. However there is a large gap between the formally specified system and the real system. Our concern in this research is to reduce this gap by mean of simulation. In this work we developed a detailed specification of the robot path planner system and we implemented the designed model as a simulation software for mobile robots.

This paper is organized as follows: In section II, the necessary background needed for describing the behavior of the path planning module is introduced; this section consists of three subsections. In the first subsection we introduce the principles of the robot navigation system. In the second subsection the principles of obstacle avoidance method is introduced. The third part is dedicated to introduce the necessary operators from process algebra theory needed for the formal specification of the system. The developed model is introduced in section III. The description of the developed simulation software and illustration scenarios are introduced in section IV. A conclusion and future work is introduced in the last section.

II. Background

A. Navigation System

In the case of mobile robots, navigation can be defined as the robot ability to act based on the environmental knowledge provided by the robot perceptual system and the goal position or list of positions in the case of a moving goal. In other words, navigation consists of moving a robot from its initial position to the destination position while avoiding obstacles on the way. In this section we give a brief description of the navigation methods introduced by [7]. The authors of this paper developed a model-based method for wheeled mobile robot navigation. The method uses the robot's kinematics equations and geometric rules. Using this method different path

978-1-4244-2794-9/09/$25.00 ©2009 IEEE

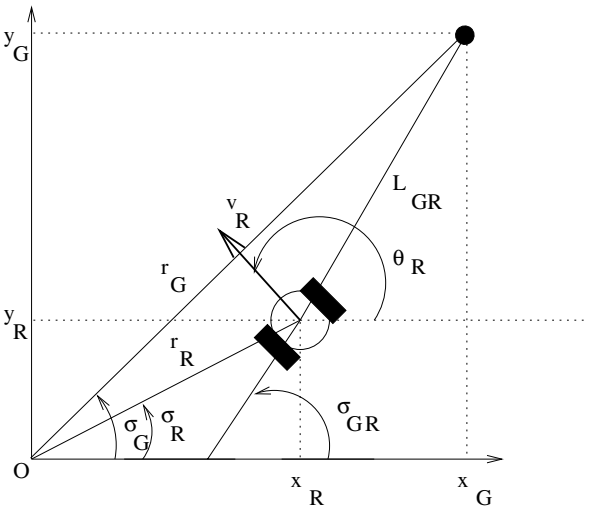

Fig. 1. The robot and the goal in the configuration space

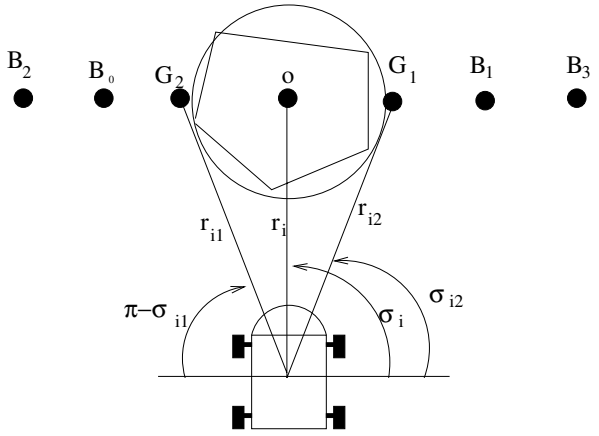

Fig. 2. A detected obstacle

can be planned by modifying the parameters of the control law. An obstacle avoidance strategy is proposed.

1) Robot Kinematics: The robot kinematics equations under the control law are described by the following equations:

$$\begin{aligned} \dot{x}_r &= v_r\ cos(N\ \sigma_{GR} + a) \\ \dot{y}_r &= v_r\ sin(N\ \sigma_{GR} + a) \\ \dot{\theta}_r &= w_r \end{aligned} \tag{1}$$

This model captures the nonholonomic constraint, which characterizes a wide range of robotic systems. The path planning method uses the proportional navigation law, which is defined in terms of the robot orientation angle as follows:

$$\theta(t) = N\ \sigma_{GR} + a$$

The geometric parameters needed for the navigation method are described in figure 1.

B. Obstacle Avoidance

Figure 2 shows a situation where an obstacle appears in the path of the robot. The robot is in collision course with the obstacle when:

$$\theta_r(t) \in [\sigma_{i1}, \sigma_{i2}]$$

The main idea to avoid the obstacle is to select the value of N and a such that

$$\theta_r = N\ \sigma_{GR} + a \notin [\sigma_{i1}, \sigma_{i2}]$$

when the robot is within a certain distance from the obstacle. In [7] the authors proposed two approaches for obstacle avoidance. The first approach, called predetermined path, is used when the geometric parameters of the obstacle, such as the center and the radius are known. The idea is to specify the value of N and a to move the robot to one of the B_i points as shown in figure 2. The second approach is used when there is no a-priori knowledge about the obstacles. The idea is to use sensor readings to construct a polar histogram, which provides obstacles directions and free directions. An intermediary goal G1 corresponding to a free direction is chosen, and the robot moves toward this point.

C. Process Algebra

The robot system is a hybrid system that exhibits both discrete and continuous behavior. In this section, a brief description of the basics, and important operators needed to describe the behavior of such system are introduced. Let P_i be denote a process and ψ a proposition then:

- The undelayable action a, is the process that performs the action a at the current point of time and then terminates successfully.
- We define the sequential composition of two processes P_1 and P_2 written $P_1.\ P_2$, as the process that first behaves like P_1, but when P_1 terminates successfully it continues by behaving like P_2. That is, P_1 is followed by P_2. If P_1 never terminates successfully, the sequential composition of P_1 and P_2 will behave like P_1.
- The alternative composition of P_1 and P_2 denoted as $P_1 + P_2$ is the process that represents the choice between the two processes P_1 and P_2. In other words, there is an arbitrary choice between P_1 and P_2. The choice is resolved on one of them performing its first action. The choice between idling processes will always be postponed until at least one of the processes can perform its first action.
- Another important process is the recursive process used to describe the infinite behavior of a system. A recursive equation is said to be guarded if it is of the form $X = a.X$ where a is an atomic action.
- The most important operator for timing a process is the relative delay denoted by $\sigma^t_{rel}(P)$. It is the process that starts executing P after the delay of t units of time.
- The relative timeout denoted by $v^t_{rel}(P)$ is the process that behaves either like the part of P that does not idle more than t units of time, or like the deadlocked process

after a delay of t units of time if P is capable of idling for t units of time.
- The relative initialization of process P denoted by $\overline{v}^t_{rel}(p)$ is the process that behaves like the part of P that idles for t units of time, if P is capable for this period of time, otherwise, it behaves like the deadlocked process after a delay of t units of time.
- The parallel composition of two process written as $P_1 \| P_2$, is the process that chooses to execute an initial action of P_1 or an initial action of P_2. The process $P_1 \| P_2$ can choose to execute a communication between P_1 and P_2. To deal with this possibility we assume a communication function $\gamma : Action \times Action \rightarrow Action$ which produces for each pair of a and b their communication $\gamma(a, b)$.
- The left merge written as $P_1 | P_1$ takes its an initial action from the process P_1 and then behaves as the merge $\|$.
- The communication merge denoted by $P_1 \mid P_2$ executes as initial transition a communication between the initial transition of the process P_1 and P_2, and then behaves as the standard merge $\|$.
- The process P proceeds conditionally on ψ, denoted by $\psi :\rightarrow P$; this means it behaves like P if the proposition ψ holds at its start.
- The process P emitting a signal ψ, denoted by $\psi \searrow P$, is the process that behaves like P and emits signal ψ.
- The process P is in evolution according to the state proposition ϕ, written $\phi \curvearrowright P$, is the process in which the emitted signal changes continuously till it performs its first action.
- The process P is in transition according to the proposition χ, written $\chi \,\vec{\cap}\, P$, is the process P in which the emitted signal changes instantaneously after performing its first action.

III. Specification of the Robot Path Planning System

1) Informal Description: The behavior of the robot planning system can be described informally as follows:

- The robot moves in four different modes using different kinematics equations. The move-to-goal mode, the move-to-intermediary-goal mode, the obstacle avoidance mode, and the path-smoothing mode. Each mode is activated/deactivated based on the information provided by the robot perceptual system. If the robot is moving in move-to-goal mode, and an obstacle is detected then the path planner determines an intermediary goal, and the robot switches to the move-to-intermediary mode. When the robot reaches the intermediary goal the obstacle avoidance mode is activated. The path smoothing mode is activated between each two modes to turn the robot in a smooth way.
- The state of the robot a time t is denoted by the vector $S_r(t) = (x_r(t), y_r(t), \theta_r(t))$.
- The desired state of the robot at the end of each motion mode is denoted by the vector $S_{rdes} = (x_{rdes}, y_{rdes}, \theta_{rdes})$.
- The desired state of the robot is attached to the process as a condition to be satisfied before switching to another motion mode.
- Before the end of the current motion mode the path planning module has to specify the next motion mode.
- In each motion mode the values of N and a are adjusted to match the desired motion mode.

2) Formal Description: For the mathematical specification of the robot path planner, we identify the following processes:
$P_{plan} = \{PATHPLANNER,\ PATHPLANNER1,$
$MOVETOGOAL, OBSTACLEAVOIDANCE,$
$MOVETOINTERGOAL,\ PATHSMOOTHING\}$
where

- $PATHPLANNER$: is the process that plans the robot motion from its initial position to the goal.
- $MOVETOGOAL$: is the process that moves the robot to the goal using the control laws.
- $OBSTACLEAVOIDANCE$: using the information from the grid map, this process consists of specifying an intermediary goal in the case of existence of an obstacle between the robot and the goal.
- $MOVETOINTERGOAL$: this process consists of moving the robot to the intermediary point specified by the $OBSTACLEAVOIDANCE$ process.
- $PATHSMOOTHING$: is the process that consists of smoothing the path between different modes of motion.

The set of atomic action can be specified as follow:
$A_{plan} = \{startrob, stoprob\}$
where

- $startrob$: stands for the action of starting the robot motion.
- $stoprob$: stands for the action of stopping the robot.

The set of atomic propositions consists of the following elements:
$P_{at} = \{intergoal,\ obstacle,\ goal,\ smoothpath\}$ where

- $intergoal$: is set to true to move the robot to an intermediary goal.
- $obstacle$: is set to true to move the robot in the obstacle avoidance mode.
- $goal$: is set to true to move the robot to the goal.
- $smoothpath$: is set to true for smoothing the path.

The behavior of the robot planning system is described as follows:
$PATHPLANNER = ((|x_R - x_G| \geq \epsilon) \wedge (|y_R - y_G| \geq \epsilon)) :\rightarrow startrob.PATHPLANNE1$
$PATHPLANNER1 =$
$(\neg intergoal \wedge \neg obstacle \wedge goal \wedge \neg smoothpath) :\rightarrow MOVETOGOAL$
$+ \ (\neg intergoal \wedge obstacle \wedge \neg goal \wedge \neg smoothpath) :\rightarrow OBSTACLEAVOIDANCE$
$+ \ (intergoal \wedge \neg obstacle \wedge \neg goal \wedge \neg smoothpath) :\rightarrow MOVETOINTERGOAL$
$+ \ (\neg intergoal \wedge \neg obstacle \wedge \neg goal \wedge smoothpath) :\rightarrow PATHSMOOTHING$

$MOVETOGOAL = (\neg intergoal \wedge \neg obstacle \wedge goal \wedge \neg smoothpath \wedge (|x_r - x_g| \geq \epsilon) \wedge (|y_r - y_g| \geq \epsilon) \wedge (\dot{x_r} = v_r cos(N\ \sigma_{GR} + a))\ \wedge (\dot{y_r} = v_r sin(N\ \sigma_{GR} + a))) \curvearrowright \sigma^*_{rel}((|x_r - x_g| = \epsilon) \wedge (|y_r - y_g| = \epsilon)) :\rightarrow (((\cdot x = x\cdot) \wedge (\cdot y = y\cdot)) \Rsh stoprob.PATHPLANNER) + (obstacle) :\rightarrow (((\cdot x = x\cdot) \wedge (\cdot y = y\cdot)) \Rsh OBSTACLEAVOIDANCE) + (intergoal) :\rightarrow (((\cdot x = x\cdot) \wedge (\cdot y = y\cdot)) \Rsh MOVETOINTERGOAL) + (smoothpath) :\rightarrow (((\cdot x = x\cdot) \wedge (\cdot y = y\cdot)) \Rsh PATHSMOOTHING))$

$MOVETOINTERGOAL = (intergoal \wedge \neg obstacle \wedge \neg goal \wedge \neg smoothpath \wedge (|x_r - x_{g_{int}}| \geq \epsilon) \wedge (|y_r - y_{g_{int}}| \geq \epsilon) \wedge (\dot{x_r} = v_r cos(N\ \sigma_{GR} + a))\ \wedge (\dot{y_r} = v_r sin(N\ \sigma_{GR} + a))) \curvearrowright \sigma^*_{rel}((|x_r - x_{g_{int}}| = \epsilon) \wedge (|y_r - y_{g_{int}}| = \epsilon)) :\rightarrow (((\cdot x = x\cdot) \wedge (\cdot y = y\cdot)) \Rsh PATHPLANNER) + (obstacle) :\rightarrow (((\cdot x = x\cdot) \wedge (\cdot y = y\cdot)) \Rsh OBSTACLEAVOIDANCE) + (goal) :\rightarrow (((\cdot x = x\cdot) \wedge (\cdot y = y\cdot)) \Rsh MOVETOGOAL) + (smoothpath) :\rightarrow (((\cdot x = x\cdot) \wedge (\cdot y = y\cdot)) \Rsh PATHSMOOTHING))$

$OBSTACLEAVOIDANCE = (\neg intergoal \wedge obstacle \wedge \neg goal \wedge \neg smoothpath \wedge (\dot{x_r} = v_r cos(N\ \sigma_{GR} + a)) \wedge (\dot{y_r} = v_r sin(N\ \sigma_{GR} + a))) \curvearrowright \sigma^*_{rel}((\neg obstacle) :\rightarrow ((\cdot x = x \cdot \wedge \cdot y = y\cdot) \Rsh PATHPLANNER) + (intergoal) :\rightarrow ((\cdot x = x \cdot \wedge \cdot y = y\cdot) \Rsh MOVETOINTERGOAL) + (goal) :\rightarrow ((\cdot x = x \cdot \wedge \cdot y = y\cdot) \Rsh MOVETOGOAL) + (smoothpath) :\rightarrow ((\cdot x = x \cdot \wedge \cdot y = y\cdot) \Rsh PATHSMOOTHING))$

$PATHSMOOTHING = (\neg intergoal \wedge \neg obstacle \wedge \neg goal \wedge smoothpath \wedge \dot{x_r} = v_r cos(N\ \sigma_{GR} + a) \wedge \dot{y_r} = v_r sin(N\ \sigma_{GR} + a)) \curvearrowright \sigma^*_{rel}((\neg smoothpath) :\rightarrow ((\cdot x = x \cdot \wedge \cdot y = y\cdot) \Rsh PATHPLANNER) + (intergoal) :\rightarrow ((\cdot x = x \cdot \wedge \cdot y = y\cdot) \Rsh MOVETOINTERGOAL) + (goal) :\rightarrow ((\cdot x = x \cdot \wedge \cdot y = y\cdot) \Rsh MOVETOGOAL) + (obstacle) :\rightarrow ((\cdot x = x \cdot \wedge \cdot y = y\cdot) \Rsh OBSTACLEAVOIDANCE))$

IV. Simulation Software

A. Implementation

As a part of this research we developed a 3D simulation software for wheeled mobile robots using Visual C++ and OpenGL. This software is the implementation of the mathematical description provided in the previous section. In this section we give a brief description of the classes developed to implement the simulator.

- CScene: this class is used to implement the robot environment. It is used to setup the OpenGl viewing volume, the projection and the camera position.

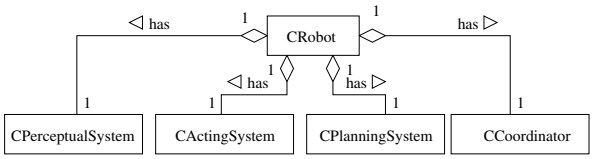

Fig. 3. The class robot

- CRobot: this class implements the simulated robot. It is used to implement the necessary methods and attributes used to draw and plan the robot motion. this class has an instance of the class CPerceptualSystem, an instance of the class CActingSystem, an instance of the class CPlanningSystem, and one instance of the class CCoordinator. The relations among these classes are shown in figure 3.
- CPerceptualSystem: this class is used to implement the robot perceptual system. It has an array of 18 instances of the class CSensor, and one instance of the class CMapBuilder. Figure 4 shows the relation between these classes.
- CActingSystem: this class implements the behavior of robot acting system. It contains four instances of the class CWheel.
- CPlanningSystem: this class contains all the functions used for planning the robot motion.
- CCoordinator: this class is used to coordinate the tasks of the perceptual system, acting system, and planning system.
- CSensor: this class simulates the behavior of the sonar sensors.
- CMapBuilder: this class contains all the functions and attributes necessary for map building. These include geometrical calculations like determining if a point is within the sonar beam or not. It contains an instance of the class CMap used to store the built map. It is also responsible for updating the map.
- CMap: this class is used to handle the storage of the map. It is used also to handle the sizing of the map to whatever size required.
- CMotionPath: this class is used to draw the robot path after each experiment.
- CTool: this class implements the needed mathematical functions, such as lines intersection, distance computation, and other functions

B. Test Cases

- Simulation 1: $x_r = 0$, $y_r = 0$, $v_r = 7$, $\theta_r = 30$, $x_g = 5$, $y_g = 50$. Figure 5 shows the initial state of the system. The results of the simulation are shown in figure 6. The dialog box in figure 7 shows the robot motion modes.
- Simulation 2: This simulation shows the robot navigating among obstacles. $x_r = 0$, $y_r = 0$, $v_r = 7$, $\theta_r = 0$, $x_o = 20$, $y_o = 50$, $x_g = 20$, $y_g = 80$. Figure 8 shows the initial state of the system, and figure 9 shows the robot path, and figure 10 shows the robot motion modes.

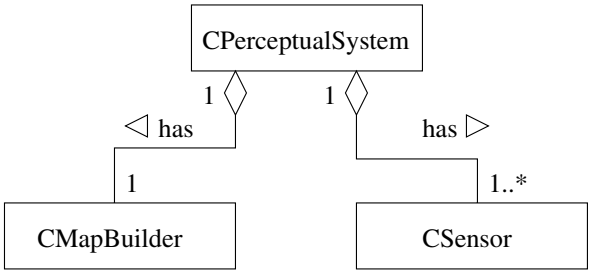

Fig. 4. The class perceptual system

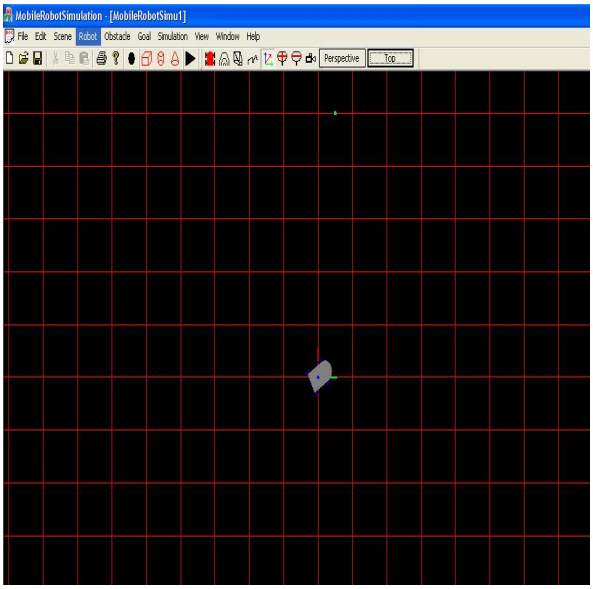

Fig. 5. The robot-goal initial position

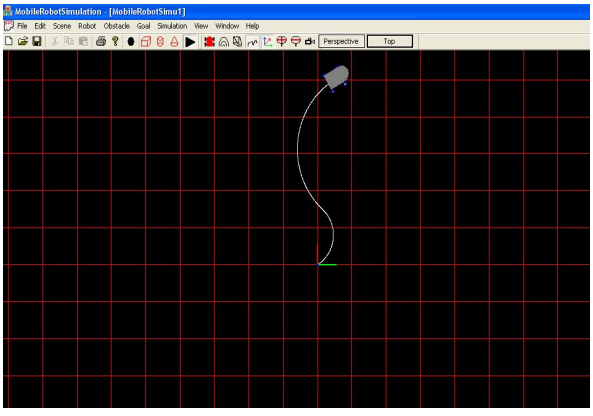

Fig. 6. Simulation results

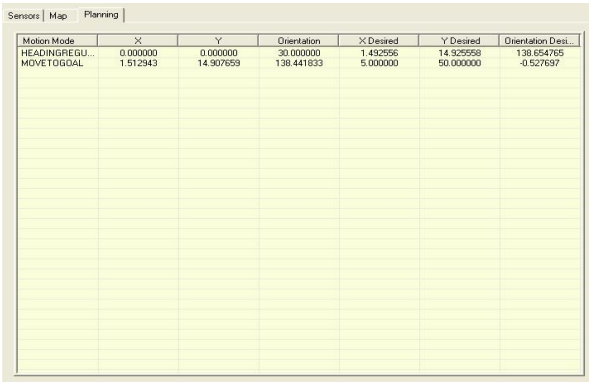

Fig. 7. Robot motion modes

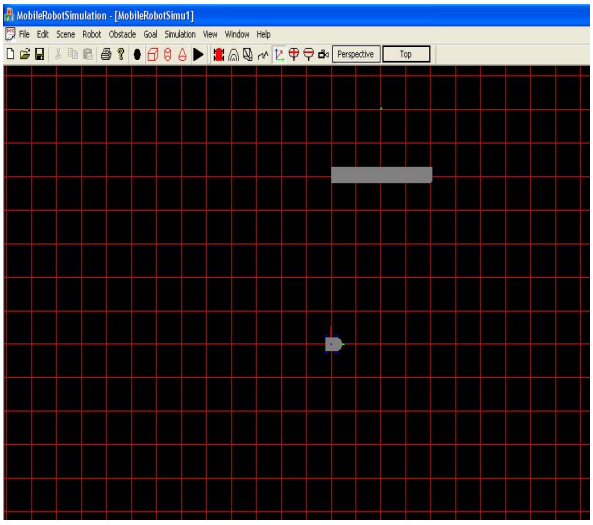

Fig. 8. The robot-goal initial position

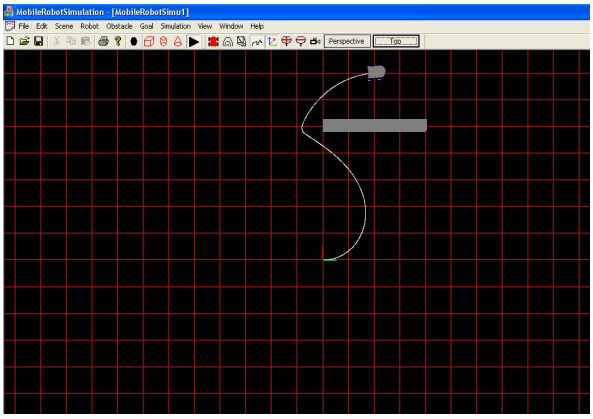

Fig. 9. Simulation results

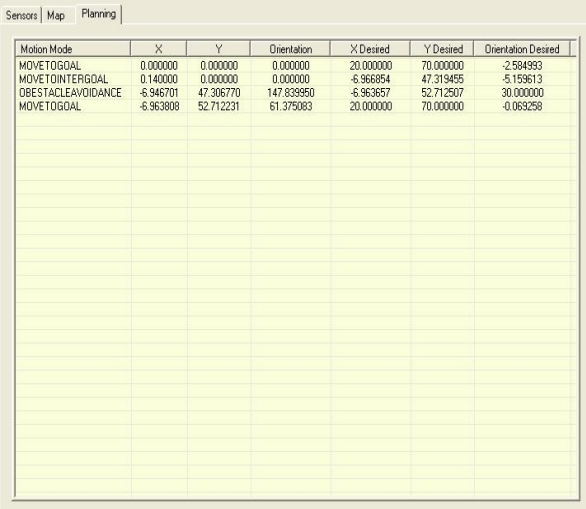

Fig. 10. Robot motion modes

V. Conclusion

In this research, we developed a detailed specification of the behavior of an autonomous wheeled mobile robot using hybrid process algebra. The robot system is divided into three main subsystems: the perceptual subsystem, the planning subsystem, and the acting subsystem. We developed a formal description of the path planning subsystem. The robot path planner is described by a powerful recursive process. The robot moves in four modes heading-regulation, move-to-goal, move-to-inter-goal, and obstacle-avoidance mode. Each motion mode is described by a process. The overall behavior of the path planner is the result of the communication between the four processes. The developed model is implemented using C++ and Opengl. In the future work we will consider the integration of the perceptual system and the acting system to derive a full description of the behavior of the robot system.

References

[1] T.A. Henzinger, "The theory of hybrid automata," in *Proceeding of the 11th Annual IEEE Symposium on Logic in Computer Science(LICS 1996)*, 1996, pp. 278–292.

[2] R. Alur and D.L. Dill, "A theory of timed automata," *Theoretical Computer Science*, vol. 126, pp. 183–235, 1994.

[3] J.A. Bergstra and C.A. Middelburg, "Process algebra for hybrid systems," *Theoretical Computer Science*, vol. 14, no. 6, pp. 215–280, 2005.

[4] P.J.L. Cuipers and M.A. Reniers, *Hybrid process algebra*, Computer Science Report 03-07, Departement of Mathematics and Computer Science Eindhoven University of Technology, July 2003.

[5] J.C.M. Baeten, *Applications of Process Algebra*, Cambridge University Press, May 22, 2004.

[6] J.C.M. Baeten and J.A. Bergstra, "Process algebra with propositional signals," *Theoretical Computer Science*, vol. 177, pp. 381–405, 1997.

[7] Fethi Belkhouche and Boumedien Belkhouche, "Wheeled mobile robot navigation using the proportional navigation," *Advanced Robotics*, pp. 1–26, 2007.