A Collaborative Approach to Solving Logistics Tasks By Mobile Agents

Guilherme Cano Lopes [1], Mateus Mussi Brugnolli, Rafael Ribeiro Nogueira Junqueira, Rafael Santos, Felipe Guerra Soares, Carlos Alberto Vilaça Jr., Marilza Antunes de Lemos [2]

Team RoboSamba, Control and Automation Engineering Department, Sao Paulo State University – UNESP, Sorocaba, SP, Brazil.

([1] contact: guilhermecano@grad.sorocaba.unesp.br, [2] Ph.D. Assistant Professor: marilza@sorocaba.unesp.br)

Abstract. Team RoboSamba research industrial logistics using Robotino with open source softwares such as ROS, OpenCV and Boost. Each team member examines specific areas of research, aiming to implement AI algorithms for creating an application with better performance in robotics competitions and further their use in industries. This application utilizes a task planner, based on the recognition of the current state of the domain, indicates what actions the robot must perform, later it calculates which is the best path to go through and how this movement should be done using a generator of trajectories.

Keywords: Mobile Robot, Planning, ROS, Robocup, Industrial Logistic, Robotino.

1. Introduction

The execution of industrial logistics using mobile robots involves two fundamental problems: cooperation and automatic planning of the tasks. The cooperation in mobile robotics defines the problem of path planning and coordinated actions where apart of avoiding collisions, robots must cooperate and keep the restrictions imposed by the task and other robots, for achieving their goals [8]. The majority of approaches for multi-robot coordination and control can be classified as deliberative or reactive, centralized or decentralized, as well as combinations of these approaches.

Control architectures for collaborative robots treat the problem as much as the application of task allocation. The problem concerns the application roles that robots can play in achieving the overall goal of the system. As task allocation with respect to the distribution of roles for each robot taking into account their availability, specialty and location [2]. Thus, the main problem lies in planning activity.

Automatic planning is a sub-field of Artificial Intelligence (AI) that studies how to implement the reasoning process capable of organizing a set of actions to achieve established objectives in a computer

[5]. Applying the results of this research area is suitable for systems that require autonomous and deliberative behavior. Some forms of planning required for automatic execution of logistics for industrial mobile robots are:

- Task planning - involves decomposition of the task into sub-tasks, ordering plans and actions, and task allocation.
- Path and drive planning - involves the resolution of a geometric task from an initial position to a target position, based on the model of the environment as well as the dynamics and kinematics of the mobile robot.
- Planning of perception - involves generating sensing action plans, for example for identification of an environment, an object and the location of a mobile robot.
- Navigation Planning - combination of motion and perception planning for following a path without colliding with obstacles until it reaches the final position.
- Communication planning – creation of dialogs for cooperation problems between some human or artificial agents.

Considering Team RoboSamba's frst experiences on latest LLSF competitions and the complexity of the proposed task, the recently team's researches focus primarily on adopting a methodology of project that considers the advance on artificial intelligence and available code reusing in the field of robotics. In this case, the Robot Operating System (ROS) was adopted by RoboSamba as a base for developing a solution for the task of LLSF competition.

ROS is indeed called a meta-operating system. It shares characteristics with some middleware systems and frameworks, but it also has OS-like features (hardware abstraction, package management, developer tool chain). In brief, ROS provides the following resources [11]:

- Publish-subscribe messaging infrastructure designed to support the quick and easy construction of distributed computing systems.
- An extensive set of tools for configuring, starting, introspecting, debugging, visualizing, logging, testing, and stopping distributed computing systems.
- A broad collection of libraries that implement useful robot functionality, with a focus on mobility, manipulation, and perception.
- ROS is supported and improved by a large community, with a strong focus on integration and documentation.

In ROS there is a Robotino Package that offers specifics resources for controlling Robotino [10]. Among them there are functions for map generation by laser rangefinder and a path planner using the generated map. Besides, libraries as Boost [2] and OpenCV [7] can be integrated on ROS. Those libraries are useful to the project for making math, logical and graphical functions (boost) and computational vision (OpenCV).

2. Materials and Methods

2.1 Materials

Hardware
- 3 Robotinos
- 3 Access Points
- 1 PC

Software/Tools
- ROS (Robotic Operating System)
- Linux OS
- itSimple
- Planners
- OpenCV Library
- API Robotino Library
- Boost Library

3. Methodology

Considering the various research fronts to be exploited to solve the problem proposed by FLC2013, team Robosamba was split so each member would focus a specific area of development. Initially, the team researched several frameworks that support development of applications for mobile robotics and as already exposed, ROS system has been selected to support research and development on industrial logistics using Robotinos. In terms of team's proposed control architecture, it is based on a centralized architecture which the control application resides on a PC that communicates with the robot via wireless network. Regarding the organization of the intelligence system, we adopted the hybrid paradigm that combines the reactivity of behavioral paradigm with the resolution of the hierarchical paradigm [Murphy, 2000]. In the hybrid paradigm, the flow of information occurs in the order "Plan-Sense-Act", in which planning is done separately, while the sensing and action are made concomitantly. In this paradigm there is still a relationship between primitive "Sense-Plan", allowing the planning to take into account updated information about the environment [1]. Figure 1 outlines the architecture of the proposed system.

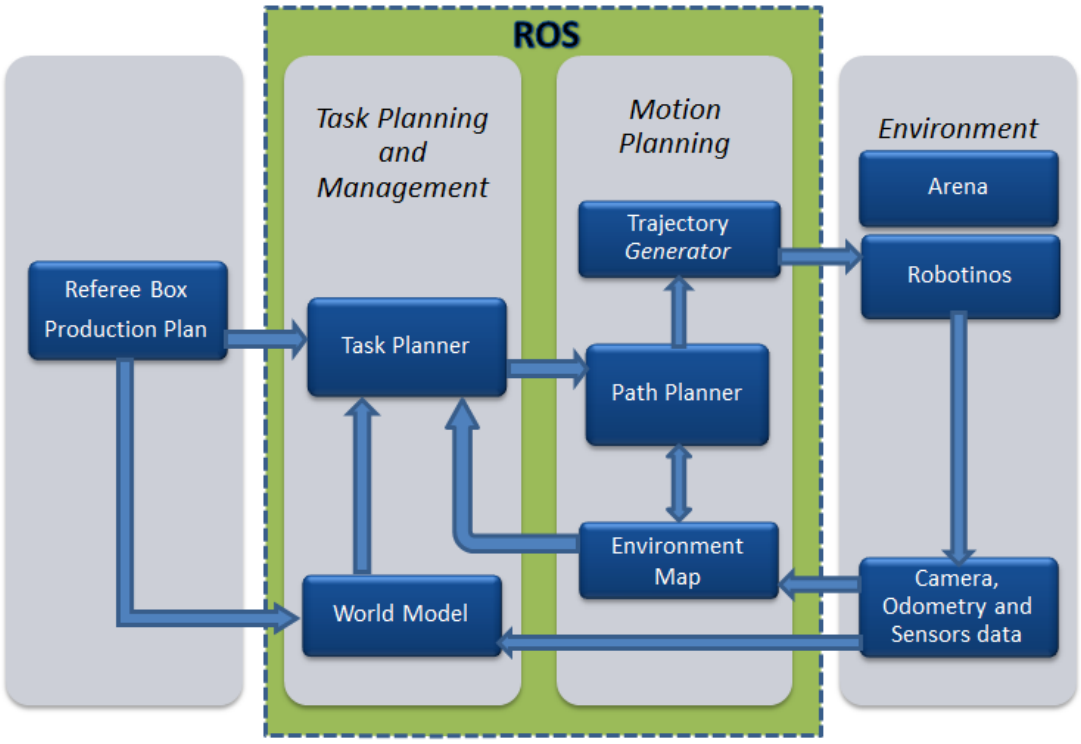

Fig. 1. System Architecture

4. World Model

Planners work manipulating information named "domain model" or "world model", which is constituted of actions and their effects that can be combined to describe possible states in the domain. For modeling the domain described on the rules of LLSF, the tool itSIMPLE [12] has been selected for planning the domain through an object-oriented language, the Unified Modeling Language (UML).

itSIMPLE also provides a mechanism to convert into PDDL language and Petri Nets. The PDDL language is appropriate to describe planning domain and is understood by several planning algorithms. Therefore the model can be simulated, tested and evaluated with a variety of planners that feature compatibility with itSIMPLE. Additionally the model obtained in Petri Nets can also be evaluated in simulators to Petri Nets [9].

5. Task Planning

One member of the team is responsible for researching and evaluating task planners, and one of those will be integrated with ROS. For this selection is necessary to take into account ease of integration into the ROS as well as possibilities in itSIMPLE, since planners as METRIC FF, GS PLAN, MAX PLAN, MARVIM and others may be integrated into it to test the model of the environment. Due to characteristics of the environment of competition, the classical planning approach will be taken, which considers the following characteristics of the environment: (i) Observable, (ii) Deterministic, (iii) Finite and (iv) Static.

6. Motion Planning

Navigation involves three basic tasks: mapping, path planning and direction. The task planning (process of highest level) specifies the input data to path planning such as destination, the agent (Robotino) and restrictions for the system.

The first part of the navigation, mapping, is performed through use of pre-stored maps or maps shaped by sensory system as the robot travels on the environment. The modeling of the environment is performed by analysis of the sensory data for the construction and modification of maps. Despite the SLAM technique has already being implemented and available in ROS, this will not be applied in solving the problem proposed. This is due to the fact that the map of the environment is static and known, by definition in the rules of the competition. Therefore, there is no need for simultaneous localization and mapping. Instead, an environment map is previously built.

The second part of navigation, path planning, is to determine possible routes on the map of the environment. The best alternative should be chosen to meet the constraints imposed by the task. Due to the characteristics of the environment of FLC2013, we chose to use Dijkstra's algorithm [4].

The third task performed in the navigation is a control movement of the robot along the path defined above, a process called trajectory generator. The movement of the robot is then controlled using its kinematic and dynamic model. During the movement of the robot, the perception process continuously scans sensory data to detect unexpected collisions.

7. Environment Map

The computer representation of the competition arena for the Dijkstra algorithm is based on using graph functions of the library Dijkstra Shortest Paths [4]. The essential parameters for the construction of the model are: number of points, height and width of the map, presence or absence of adjacencies between points and their equivalent costs, if any. All such data is stored in a graph structure that represents the competition's arena. It is possible to indicate the path chosen by the algorithm using different colors, for example, use white color for unaffected points, gray color to points that can be chosen in the trajectory and black to indicate the best path.

8. Path Planning

As previously cited, for path planning the team opted for using Dijkstra's algorithm. This algorithm is applied in the following way: It is defined that every point has its own internal cost or distance equal to infinity, except the origin point, which costs zero. Then, the algorithm calculates paths from origin to every adjacent point, adding the cost of the way-out point plus the path cost. By doing this successively and considering that at least one path that links the origin point to the desired point exists, the algorithm informs the less expensive path. The Dijkstra algorithm has strategies for avoiding infinite trajectories and high cost paths.

The trajectories cost is defined when building a graph that represents the environment map. However it is interesting to change some costs before the path planning. For avoiding collision of two or more Robotinos, an exclusive way is assigned to each robot that is about to execute a move through a path. Therefore, while an exclusive trajectory is being used, the Dijkstra's algorithm chooses alternative paths for the remaining robots.

In order to change costs in a controlled way, the Heap library [6] was used. This library is able to organize a priority line which its contents order is related to its cost. This way, it is possible to pick a preferential path for a task. The priority line is generated as there is need to create exclusive trajectories.

9. Trajectory generation

The trajectory generation is one of the essential building blocks of a motion planner for autonomous mobile robots. It consists of determining a relationship between an admissible path and time so that this information can be used by the control system of the robot. The path planner determines a set of positions and orientations avoiding collisions with obstacles in the workspace. The trajectory generator lists the elements of the path with time. Finally, the control system, responsible for the execution of the trajectory, generates a set of forces and torques to the actuators of the robot, so that its movement in the workspace is performed. This is a complex problem that is still under investigation. The idea is to transform poses set (from algoritmo Dijkstra) in a continuous path in terms of linear velocity (Vx) and angular velocity (ω). The orientation is defined by the alignment of two subsequent grids and can be 0°, 90°, 180° or 270°. When a Robotino moves from a grid to the next in a linear trajectory, then $Vx = K_{line}$ (arbitrary velocity constant) and $\omega = 0$. The time necessary for realizing a completely linear trajectory (t_{line}) between grids is:

$$t_{line} = l/v$$

At curves, the robot's trajectories are a quarter of circumference where $Vx = K_{line}$ and $\omega = K_{angle}$. The time necessary for completing a curve trajectory is:

$$t_{angle} = 2\pi l/v$$

and:

$$K_{angle} = \pi/2t_{angle} = v/4l$$

The team is currently testing a software platform called TAO, which can be used for different types of robots and can be adapted for Robotino API2 in order to use the path control library. Depending on the results, it may be used for this purpose in LLSF this year.

10. World Perception

The computer vision systems have been widely used in robot navigation tasks. Since Robotinos are provided with cameras, image processing techniques through OpenCV Library are being used for detection of types of machines, machine states and identification of pucks. Additionally, infrared sensors, odometer and gyroscope feed system aiding in the detection of the current state of the world.

A few changes were made to the Robotino's hardware: a new sliding unit was developed to suit completely rotational movements without losing of the puck. The new sliding unit sensors were changed for two optical sensors for more reliability when adjusting the position of the robot for aligning it to the machines.

11. References

1. Alves, S. F. R. Plataforma de software para técnicas de navegação e colaboração de robôs móveis autônomos. Dissertação de Mestrado. Universidade Estadual de Campinas, Faculdade de Engenharia Mecânica. Campinas, SP, 2011.

2. ARAI, T.; PAGELLO, E.; PARKER, L. E. Editorial: Advances in multi-robot systems. IEEE Transactions on Robotics and Automation, v. 18, n. 5, p. 655–661, 2002.

3. Boost. Boost Library.
http://www.boost.org

4. Dijkstra. Dijkstra Shortest Paths.
www.boost.org/doc/libs/1_53_0/libs/graph/doc/dijkstra_shortest_paths.html

5. Ghallab, M., Nau, D. and Traverso, P. Automated Planning: Theory and Practice Morgan Kaufmann Publishers, 2004.

6. Heap. Heap Library.
http://www.boost.org/doc/libs/1_53_0/doc/html/heap.html

7. OpenCV. OpenCV Library
http://opencv.org.

8. Pereira, G. A. S. Navegação e Controle de Robôs Móveis Cooperativos: Uma Abordagem Baseada em Conectividade de Grafos. *PhD Thesis Universidade Federal de Minas Gerais*, 2003.
http://www.verlab.dcc.ufmg.br/projetos/gpereira/thesis

9. PetriNets. Petri Net World.
http://www.informatik.uni-hamburg.de/TGI/PetriNets/

10. Robotino. Package Robotino no ROS.
http://www.ros.org/wiki/robotino

11. ROS. Robot Operation System.
http://www.ros.org/wiki

12. Vaquero, T.; Tonaco, R.; Costa, G.; Tonidandel, F.; Silva, J. R. and Beck, J. C. itSIMPLE4.0: Enhancing the Modeling Experience of Planning Problems. In 22nd International Conference on Automated Planning and Scheduling (ICAPS2012). Atibaia, Sao Paulo, Brazil, June 25-29th, 2012.