RESCUE
Cooperative Navigation for Rescue Robots
2nd Year Technical Report

A. Bernardino, L. Custódio, J. Frazão,
T. Krause, P. Lima, M. I. Ribeiro,
J. Santos-Victor, A. Vale

July 17, 2003

1 Introduction

This document is a slightly more detailed report of the work executed during the second year of the RESCUE project. In Section 2 the topological navigation of the land robot, in its map building and localization facets, is covered, including experimental results in outdoors environments. Most of the work described in this section was carried out by the project members Alberto Vale and Isabel Ribeiro. In Section 3 the vision-based navigation of the aerial blimp robot is described, namely the development of a motion library to support future developments. Most of the work described in this section was carried out by Thomas Krause, a German student who spent 4 months at ISR/IST working in the project, Alexandre Bernardino and Pedro Lima. Section 4 refers to the design and first steps towards implementation of the software and functional architectures that will support the project development. Most of the work described in this section was carried out by the project member João Frazão.

2 Topological Navigation of Land Robot

2.1 Map Building and Localization

To support the navigation of the land robot a topological map was developed, which can be achieved by a probabilistic approach on the world representation. The robot perception is condensed in observations, o_t, that represent the information resulting from the processing of the raw data acquired at each time instant t.

An observation is a vector where each component relates to a different feature. This perception has to be recorded in a map, that is composed by a set of states $S = \{s_i\}$. Each state, containing a partial representation of a physical area of the environment, is characterized by a set of relevant features that support the state identification and prevent state mismatching. According to the uncertainty of the measurements, each map's state s_i is represented by a Gaussian pdf. With this map characterization, the mapping procedure consists in estimating the mean vectors and the covariance matrices that maximize the probability of all observations given the environment model, i.e., that maximizes the likelihood function. As this maximization is a hard problem to solve, a way to overcome the associated computational burden is by changing the corresponding likelihood function by $F(S)$ in (1), the expectation of the likelihood given a previous estimation of the model, [6], [7]. The maximization of (1) is performed through a modified version of the Estimation and Maximization algorithm.

$$F(s) = E\left\{\log(p(O \mid S)) \mid S^{old}\right\} \tag{1}$$

The robot estimated location at time instant t, $\hat{q}_t$, is the map state that most likely produced the observations acquired by the robot sensors during a given time interval, T. The proposed localization procedure yields a robot estimated location, $\hat{q}_t = s_i$; note that this does not mean that the robot physical location (pose) coincides with that of the environment place that lead to the map state s_i. According to a probabilistic approach, the current estimated location, $\hat{q}_t$, is the argument that maximizes the pdf of the location given the observation sequence $O_T = \{o_1, o_2, \ldots, o_t, \ldots, o_T\}$ acquired in the time interval T, i.e.,

$$\hat{q}_t = \arg\max_{q_t} P(q_t = s_i \mid o_1, \ldots, o_t, \ldots, o_T). \tag{2}$$

Based on Markov Models, the localization procedure in (2) is similar to the high-dimensional maximum likelihood estimation problem. This problem is efficiently solved using a modified version of the Forward-Backward (FB) algorithm, ([7], [9]), described in [4].

2.2 Feature Extraction

Topological maps provide useful abstractions of an environment, showing natural features that characterize particular locations or places. The algorithm developed in the frame of the Rescue project is intended to adapt to the available sensors, this meaning that adding or removing different types of sensors enlarges or reduces the number of properties available to the algorithm. The raw data provided by the sensors requires a signal processing procedure before the implementation of the feature extraction. In the present work, the available sensors provide position (from odometry), orientation (including GPS), range (from a laser scanner and sonars) and intensity information (from a vision camera). The main features considered at the actual stage of the project are the free-area in front of the robot and its variance (from laser and sonars sensors) for indoors experiences and image histograms and orientation for outdoors experiences. The features are selected according to the target scenario, mainly in outdoor environments. If the features are not well selected, in spite of a good representation, the mapping algorithm provides ambiguities over the representation. Therefore, it is also necessary to define a criteria to analyze the features quality based on the measurements, [2]. The current research focus the feature extraction from the laser sensor and from intensity images, which is a strong source of information. Examples of features are: color, geometric forms, histograms, orientation, pattern, from image and principal components from the Laser, as referred in [3] and [8].

2.3 Experimental Results

The mapping algorithm was tested in outdoors environments, using the sensor information acquired by the ATRV-Jr rover during the path execution. The scenario coped a large area (approximately $1500m^2$) represented in Figure 1, involving buildings, trees, cars parked and moving and people walking. The robot was completely autonomous when following a path defined by a set of via points. Each via point was parameterized by the GPS coordinates. To accomplish this mission, two behaviours based on [5] were implemented: one to follow the via points and the other to achieve obstacle avoidance. To face a possible temporarily GPS failure, a Kalman Filter based approach implemented dead-reckoning. The feature extracted from the intensity data, recorded by the robot during the path following, was the brightness histograms. Using this feature, the mapping algorithm produced a topological map defined by a set states. There might be ambiguity in a couple of states, (see the dashed line in Figure 2), since they correspond to different geographical areas, different images, but with similar brightness histograms. Consequently, the type and number of features has to be correctly selected to avoid mismatching.

Figure 1: The scenario

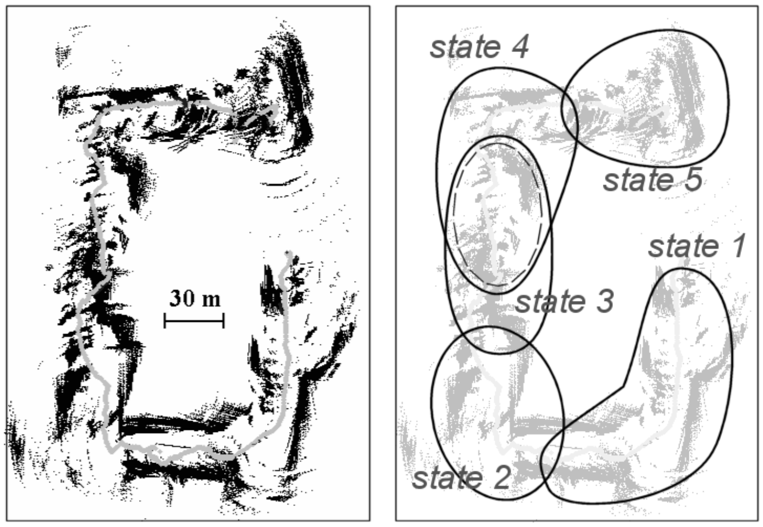

Figure 2: Left: The Laser and sonar measurements acquired in the environment. Right: The Topological map compiled by the algorithm

The future work includes a deep choice and selection of the features and the integration of the Markov Model approach presented in [4] for localization in a simultaneous localization and mapping procedure (SLAM) aiming a topological navigation at search and rescue operations.

3 Vision-Based Aerial Blimp Navigation

The aerial blimp robot is now able to fly. The first step consisted of developing a driver to provide access to the radio controller, whose hardware had been modified in the first year to allow remote control by a ground computer. Then, several measurements were made to obtain the characteristic functions that relate the voltage sent to the motors with the thrust generated by the propellers. All resulting curves and fitted functions are available. Using that, a control library for position and velocity control was developed. The library is modular and consists of three control levels (see Figure 3), from position control in world coordinates down to velocity control of forward/backward, upward/downward and rotation around the vertical axis movements, in vehicle coordinates. This control library can be used like a black box. The user can control separately the local velocities over the global direction control up to the global position control [1].

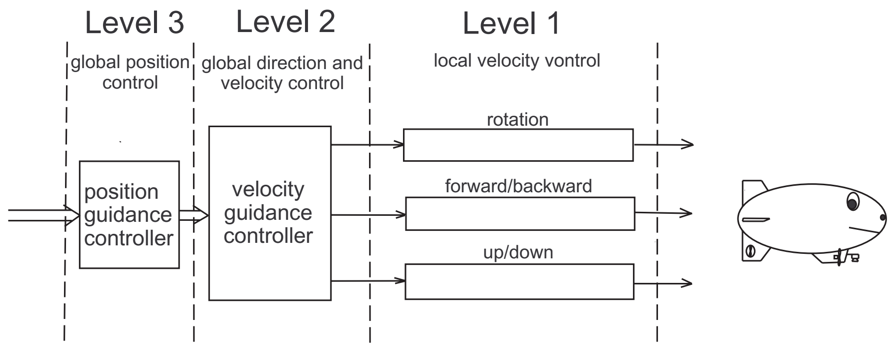

Figure 3: main control structure

Feedback for the closed-loop controller is provided by a video camera, the single on board sensor avilable. The optical flow is calculated from image sequences and used as an estimator of the blimp velocity components. The integration of the optical flow provides rough position estimation.

Several indoors tests were performed and their results will soon be available in a technical report. Generally, the results were positive, and it is possible to control the robot velocity in 3D space, including the compensation of dynamical effects like simulated weak wind. Nevertheless, optical flow has some drawbacks, especially notable after integration, resulting in an often poor odometric estimate of the blimp position in world coordinates.

Plans for future work include the development of visual tracking algorithms to enable the aerial blimp following the land robot, and to provide a vision-based topological map of the scene below the blimp, as specified in the project reference scenario.

4 Software and Functional Architectures

The first part of the work carried out in the second year concerning architectural aspects was concerned with concept definitions and specification. We define a Rescue Agent as an agent with its own execution context, its own state and memory and a way of sensing and taking actions on the environment. Each of the agents is an active object with two defined ports in the upper interface. One of the ports is the input port and one can see it like the sense port from where the agent is notified of changes in the world. Another can see it like the request port from where the agent receives notifications of actions to perform from higher-level agents. The other port is the output port from where the agent reports progress to the caller or from where the agent throws events to higher-level agents. This is what we define as the consistent interface for communication and control. The agent can also have a lower level interface from where he can control and sense the agents beneath him. The lower level interface is customized in accordance to the type of agent, for instance an agent finite state machine has so many lower level control ports as agents that he is controlling and a lower level input port where all lower level agents write events. The ports are linked together through the blackboard. For configuration flexibility of the agent's hierarchy, the agent upper ports are never assigned in the definition of the agent. Ports are assigned in the definition of the mission. Ports are really a synchronized data entry on the blackboard.

Each agent also defines a new scope (his scope) inside of the blackboard. This scope can be viewed as the memory of the agent. Different agent types are supported under this concept [1]. The combinations between different agent types provides the flexibility required by a functional architecture as that of the Rescue project. For special interactions that are not supported yet, the architecture is open in a way to include other types of agents.

The architecture consists of three basic components (agents, blackboard and control/communications interface), as well as of five Execution Modes for each of those components: the Control Mode coordinates the run-time interactions between the basic components, while the Design, Calibration, Supervisory Control, and Logging and Data Modes concern the programmer interface.

The control from an upper agent to a lower agent is done trough special and well-defined functions: start, stop, set and reset. In this sense if we stop the agent that mimics the fleet, it will request his agents (the robots) to stop, so a cascading reaction will stop all the agents' hierarchy inside each of the robots, from the top until the low-level hardware agents. Similar behaviour happens with the start command.

After the definition of the software and functional architectures, current work is focussed on its implementation, namely on the implementation of the distributed blackboard and some basic agents. Right now, we are focusing mostly on the Control Mode of the architecture.

The blackboard ports are already implemented, as special FIFO (first in first out) channels, through which the agents can write or read samples.

Samples are a special type of data that have coupled a time tag and a sequence number given at data creation time. The data inside a Sample can be of several types like integer, float and char. We have also dynamical arrays of integer, float and char. For instance an array of two floats, can be used for multidimensional data like XY velocity vectors.

The blackboard ports are special FIFOs, since they have several features and support a special set of operations. The reader agent is able to get the first produced sample, the last one, or the sample with order "n". The maximum size of the FIFO is decided upon the declaration of the port. The size of the FIFO is preserved by discarding the oldest sample. The ports have a blocking read operation and a non-blocking one. Writing is always non-blocking.

In addition to get (read from the port) the samples by the order number, an agent can get the samples by its time tage: the sample with time tag closest to the time specified is returned.

The main goal is that the agents can pick from several (input) ports the samples "simultaneously" without being blocked by the ports, then they process the samples (the loop time), and afterwards they write the result to the (output) ports. Agents do everything asynchronously. At the end of this step they read from the control port (using a non-block read) to see if they have received some message from upper level (lets say to stop). Otherwise they keep looping.

The ports have a global identifier name, and a CORBA interface in the network. This mean that agents located in different robots are able to share data. Furthermore, they are able to start and stop other agents (by issuing commands on the control ports) located in different computers.

The ports provide the glue to couple the agents together. With the added functionalities to the ports an agent is able to keep working even if is getting behind the agents that are producing the data. This approach also fits nicely for data signal processing, since the ports have a history of value and time pairs, instead of having only the last value.

Also already implemented is the simple base class for a control loop or periodical agent that is able to run a function periodically given a customizable time period. This agent has one input control port and an output control port. This agent runs its function periodically after it is told to start (message start on the control port), and stops when is ordered to stop (stop message on control port).

Blocking reads and signals are being used on the discrete part of the system, namely in the state machine class, witch is being currently implemented. The state machine is event-driven and waits until some event occurs and then changes its state. Therefore, instead of looping on the input ports until some command arrives or some condition change it is rather blocked waiting for a transition event from down level or from a message from upper level to stop.

References

[1] P. Lima, M. Isabel Ribeiro, Luis Custodio, Jose Santos-Victor "The RESCUE Project - Cooperative Navigation for Rescue Robots", Proc. of ASER'03 - 1st International Workshop on Advances in Service Robotics, March 13-15, 2003 - Bardolino, Italy.

[2] M. Law, A. K. Jain, M. Figueiredo. "Feature selection in mixture-based clustering", Neural Information Processing Systems, 2002.

[3] S. Grigorescu, N. Petkov, P. Kruizinga. "Comparison of Texture Features Based on Gabor Filters", IEEE Transactions on Image Processing, **11(10)**, 2002.

[4] A. Vale, M. I. Ribeiro. "A Probabilistic Approach for the Localization of Mobile Robots in Topological Maps", Proc. of the 10th IEEE Mediterranean Conf. on Control and Automation, Lisboa, Portugal, 2002.

[5] E. Bicho. "Dynamic approach to behavior-based robotics: design, specification, analysis, simulation and implementation", Shaker Verlag, Aachen, ISBN 3-8265-7462-1, (2000).

[6] S. Thrun. "Probabilistic Algorithm in Robotics" Artificial Inteligence Mag., **21(4)**, pp. 93–109, 2000.

[7] S. Thrun, W. Burgard and D. Fox, "A Real-Time Algorithm for Mobile Robot Mapping with Applications to Multi-Robot and 3D Mapping", Proc. of the IEEE Int. Conf. on Robotics and Automation, 2000.

[8] F. Wallner, B. Schiele and J. Crowley. "Position Estimation for a Mobile Robot From Principal Components of Laser Range Data", Robotics and Autonomous Systems, (1998).

[9] M. Kijima. "Markov Processes for Stochastic Modeling", Chapman & Hall, (1997).