Robel : Synthesizing and Controlling Complex Robust Robot Behaviors

Morisset Benoit[1] and **Infantes Guillaume** and **Ghallab Malik** and **Ingrand Felix**[2]

Abstract. We briefly present the Robel supervision system [3] which learns from experience robust ways to perform high level tasks. Each possible way to perform a task is modeled as a Hierarchical Tasks Network whose primitives are sensory-motor functions. The relationship between supervision states and the appropriate modality is learned through experience as a Markov Decision Process (MDP). This MDP is independent of the environment and characterizes the robot abilities for the task.

Presentation

Robust robot navigation is a complex task which involves many sensory-motor (*sm*) functions such as localization, path planning, terrain modeling, motion generation adapted to obstacles, and so on. Since no single method or sensor has a universal coverage, each *sm* function has its specific weak and strong points. The approach presented here improves the global robustness of complex tasks execution in taking advantage of these *sm* functions complementarity.

We propose a two-stepped approach named *RO*bot *BE*havior *L*earning. First, *sm* functions are synthetized in a collection of Hierarchical Tasks Networks (HTN) [2], that are complex plans called *modalities*. Each modality is a way to achieve the desired task. The second contribution of this work is an original approach for learning from the robot experiences an MDP-based supervision graph which enables to choose dynamically the most appropriate modality to the current context. We thus obtain a system able to efficiently use redundancies of low-level *sm* functions to robustly perform high-level tasks.

1 Synthesis of Modalities

A high level task given by a mission planning step requires an integrated use of several *sm* functions. Each consistent combination of these *sm* functions is a particular plan called a *modality*, one way of performing the task. A modality has specific characteristics that make it more appropriate for some contexts or environments, and less for others. The selection of the right modality is performed by the *controller* (see Section 2) all along the task execution.

We chose to represent modalities as Hierarchical Task Networks because of their expressiveness and their flexible control structure. HTNs offer a middle ground between programming and automated planning, thus we can use the same formalism to write a modality by hand or to generate it automatically from a few information on how the *sm* work (e.g. their inputs and outputs, the resources they use, time, synchronization...). The next challenge is to learn to use them efficiently.

2 The Controller

We present in this section an example of a controller adapted to an indoor navigation task. To perform this task, the design of the control space and the control process itself requires the use of a topological graph. Cells are polygons that partition the metric map. Each cell is characterized by features such as *Corridor, Large Door, Narrow Door, Confined Area...* The controller has to choose a modality that is most appropriate to the current execution state for pursuing the task. In order to do this, a set of *control variables* has to represent control information for the *sm* functions. The choice of these control variables is an important design issue.

For example, in the navigation task in an indoor environment, the control variables are the **cluttering** of the environment, the **angular variation** of the profile of the laser range data, the **inaccuracy of the position estimate**, the **confidence** in the position estimate, the **navigation color** (in a topological graph) and the **current modality**.

A control state is characterized by the discretized values of these control variables. We finally end-up with a discrete control space which allows us to define a *control automaton*. The control automaton is nondeterministic: unpredictable external events may modify the environment, e.g. someone passing by may change the value of one of the variables. Therefore the execution of the same modality in a given state may lead to different adjacent states.

The Control automaton Σ is a Markov Decision Process. As an MDP, Σ could be used reactively on the basis of a universal policy π which selects for a given state s the best modality $\pi(s)$ to be executed. However, such a universal policy will not take into account the current navigation goal. A more precise approach takes into account explicitly the navigation goal, transposed into Σ as a set S_g of goal states in the control space. This set S_g is given by a look-ahead mechanism based on a search for a path in Σ that reflects a topological route to the navigation goal.

Goal States in the Control Space

Given a navigation task, a search in the topological graph provides an optimal route r to the goal, which is characterized by the sequence of colors of traversed cells, and its length.

Now, a path between two states in Σ defines also a sequence of colors σ_{path}, those of traversed states; it has a total cost, that is the sum $\sum_{path} C(a, s, s')$ over all traversed arcs. A path in Σ from the

[1] SRI International, 333 Ravenswood Avenue, Menlo Park, CA 94025-3493, USA. email: morisset@ai.sri.com
[2] LAAS-CNRS, 7 avenue du Colonel Roche 31077 Toulouse Cedex 4 FRANCE email: firstname.lastname@laas.fr
[3] this paper is a short version of [1]

current control state s_0 to a state s corresponds to the planned route when the path *matches* the route (σ_r, l_r) in the following way:

- σ_{path} corresponds to the same sequence of colors as σ_r with possible repetition factors; this requires that we will be traversing in Σ control states having the same color as the planned route.
- $\sum_{path} c(a, s, s') \geq K l_r$, K being a constant ratio between the cost of a state-transition in the control automaton to corresponding route length; this condition enables to prune paths in Σ that meet the condition on the sequence of colors but cannot correspond to the planned route.

Let route(s_0, s) be true whenever the optimal path in Σ from s_0 to s meets the two previous conditions, and let $S_g = \{s \in S \mid \text{route}(s_0, s)\}$ (built using a Moore-Dijkstra algorithm starting from s_0). It is important to notice that this set S_g of control states is a *heuristic projection* of the planned route to the goal. There is no guarantee that following blindly a path in Σ that meets route(s_0, s) will lead to the goal, and there is no guarantee that every successful navigation to the goal corresponds to a sequence of control states that meets route(s_0, s). This is only an efficient and reliable way of focusing the MDP cost function with respect to the navigation goal and to the planned route.

Finding a Control Policy

At this point we have to find the best modality to apply to the current state s_0 in order to reach a state in S_g, given the probability distribution function P and the cost function C. A simple adaptation of the *Value Iteration* algorithm solves this problem. Here we only need to know $\pi(s_0)$. Hence the algorithm can be focused on a subset of states, basically those explored by the Moore-Dijkstra algorithm.

The closed-loop controller uses this policy as follows:

- the computed modality $\pi(s_0)$ is executed;
- the robot observes the state s, it updates its route r and its set S_g of goal states, it finds the new modality to apply to s.

This is repeated until the control reports a success or a failure. Recovery from a failure state consists in trying from the parent state an untried modality. If none is available, a global failure of the task is reported.

Estimating the Parameters of the Control automaton

A sequence of randomly generated navigation goals is given to the robot. During its motion, new control states are met and new transitions are recorded or updated. Each time a transition from s to s' with modality a is performed, the traversed distance and speed are recorded, and the average speed v of this transition is updated. The cost of the transition $C(a, s, s')$ can be defined as a weighted average of the traversal time for this transition taking into account the eventual control steps required during the execution of the modality a in s together with the outcome of that control. The statistics on $a(s)$ are recorded to update the probability distribution function.

3 Experimental results

The justification of the whole system relies on the following principle : the use of the complementarity of several navigation modalities increases the global robustness of the task execution. To validate this principle, 5 handwritten modalities have been integrated on board our XR4000 robot. In order to characterize the usefulness domain of each modality, we measured in a series of navigation tasks, the success rate and other parameters such as the average speed, the distance covered, the number of retries. Various cases of navigation have been considered such as for instance, long corridors or large areas, cluttered or not, occluding the 2D characteristic edges of the area or not. These extensive experiments required several kilometers of navigation. The result is that for each case of navigation met by the robot there is at least one successful modality. On the other hand, no modality is able to cover all cases. This result clearly supports our approach of a supervision controller switching from one modality to another one according to the context.

We propose here to illustrate the learning capabilities of the controller through an indoor navigation task. To perform this experiment, we start with an empty automaton and 2 complementary modalities (M_1, M_2). The learning starts with a series of 83 navigations in a large open environment. The constant selection of M_1 by π all along the 30 last navigations shows that the controller relevantly learned the superiority of M_1 on M_2 for the open environments. Some narrow obstacles are added. If in the previous phase, M_1 was more appropriate than M_2, in this second phase, the system is able to learn within 30 navigations, the better efficiency of M_2 in cluttered environments. The goal of the third step is to check if the learning of the second phase (avoidance) didn't corrupt the learning of the first phase (open navigation). The obstacles are then removed to recover the same environment as the phase 1 and 58 more navigations are performed by the system. This last step shows that the efficiency of M_1 in the phase 1 has not been forgotten after the learning of the phase 2.

Conclusion

This paper addressed the issue of producing complex modalities from sensory motors functions, and how to exploit the complementarity of these modalities to perform a task.

We would like to emphasize two particular features of our method:

Portability: Control state variables reflect control information for the *sm* functions. No information dedicated to the environment is present in the control state. Thanks to this, a controller learned in an environment can directly be used in another environment.

Adaptativity: In this system, learning and execution are not decoupled : learning of Σ parameters is active all along the robot navigations. If a new situation is encountered, corresponding new states are created and the new untried transitions are taken into account by the next computations. This unsupervised learning confers a high level of adaptativity to the controller.

In addition to future work directions mentioned above, an important test of Robel [1] will be the extension of the set of tasks to manipulation tasks such as *"open a door"*. This significant development will require the integration of new manipulation functions, the synthesizing of new modalities for these tasks and the extension of the controller state. Another development which seems rather promising is to learn the control space of the controller instead of relying on one given by hand.

REFERENCES

[1] B. Morriset G. Infantes M. Ghallab F. Ingrand, 'Robel: Synthesizing and controlling complex robust robot behaviors', in *4th International Cognitive Robotics Workshop (submitted)*, (2004). http://www.laas.fr/~felix/publis.

[2] D. Nau, Y. Caoand, A. Lotem, and H. Munoz-Avila., 'Shop: Simple hierarchical ordered planner', in *IJCAI*, (1999).