Learning Navigation Teleo-operators with Behavioural Cloning

Blanca A. Vargas Govea, Eduardo Morales Manzanares
Department of Computer Science
blanca@inaoep.mx, emorales@inaoep.mx

ABSTRACT

Programming a robot to perform tasks in dynamic environments is a complex process. The robot requires skills to react according to unexpected changes without losing its primary goal. Teleo-operators or TOPs have proved to be an effective framework for mobile robots to achieve goals in dynamic environments. However, their definition is a difficult and time-consuming process. In this paper, a system called TOPSY (Teleo-OPerator learning SYstem), that can learn TOPs from human-guided traces of simple mobile robot tasks is described. TOPSY transforms a large set of low level readings from the robot sensors into a small set of high level concepts based on natural landmarks. This information with additional background knowledge is given to an inductive logic programming system to learn other high level concepts and TOPs. It is shown how TOPSY learns how to avoid obstacles, orient towards a goal, and reach it.

I INTRODUCTION

Using robots to develop everyday tasks is an increasing trend. Tasks like elderly assistance and tourist guidance are carried out in dynamic environments where the robot has to cope with unexpected situations. Teleo-reactive programming [7] is a well suited formalism to deal with goals in dynamic environments. Teleo-reactive programs are sets of reactive rules or teleo-operators (TOPs) that sense the environment continuously and apply actions whose continuous execution will eventually satisfy the goal condition. However, programming TOPs can be a difficult and time-consuming process.

The learning of TOPs has been addressed by a few frameworks. TRAIL [1] was the first system that applied the teleo-reactive formalism. It learns TOPs from traces to control a simulated constructor robot and to fly a plane. In other framework [5], the system depends on annotated behavior traces where the user marks information about the goals and actions. Broda presents a method [3] based on the construction of an automata; the approach is propositional and it was tested in the blocks world domain. In real valued domains it is a limitation because of the huge amount of possible states that can be reached. Therefore, it is difficult to extend it to robotics. In contrast to these systems, our work is focused on real robots, domain with the additional problem of coping with sensors.

In robotics, Zelek [10] developed a control architecture for mobile robots with a visual environment based on TOPs. The limitation is that the TOPs have to be constructed manually. In this paper we present an automatic method to generate navigation TOPs for mobile robots.

A technique used to learn skills from operational traces is known as behavioural cloning [6]. In its original state-action mapping approach, the resulting clones were not robust to changes in the control task, their representation lacks structure [2] and handling goals is not straightforward. Our behavioural cloning approach learns sub-tasks and combines them in a teleo-reactive framework.

In this paper, a system called TOPSY that is able to learn navigation teleo-operators from traces of robots is described. TOPSY uses traces of sensor's output of the robot performing a task. From this raw data, TOPSY identifies natural landmarks, such as walls, corners and discontinuities. This information is transformed into a relational representation that is given with additional background knowledge, to an ILP system to learn either new background knowledge relations or new TOPs. In this paper, we use Aleph [8] as our ILP system. TOPSY is able to learn useful concepts for mobile robots like safe zones to turn, and several TOPs expressing primitive skills, like obstacle avoidance and orientation towards a goal, and then used them to safely go to a goal in a dynamic environment. This paper is structured as follows. Section II presents an overview of TOPSY. Section III describes the natural landmark representation. Section IV explains how TOPSY learns relational concepts and TOPs. In section V, experiments and results are described. Finally, conclusions and future research directions are given in Section VI.

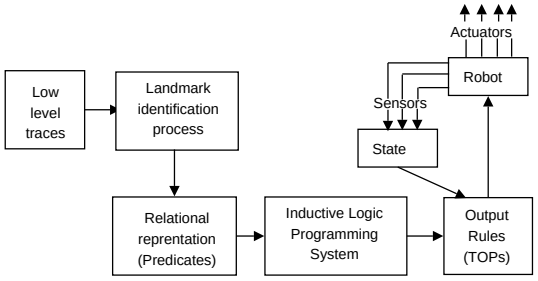


Figure 1: Block diagram

II An Overview of TOPSY

This section briefly describes the main components of TOPSY as shown in Figure 1.

- Information from the sensors of a mobile robot are given to TOPSY. We used information from a laser sensor and from a ring of sonars.
- A natural landmark identification process is used to obtain a small set of high level information of the environment.
- A pre-processing module transforms the input information into facts and generates the files needed to the learning process.
- Information about the landmarks, with possibly additional background knowledge, is used to produce a relational representation of the environment.
- The relational information is given to an ILP system to induce new relations and new TOPs.

These parts are detailed in the following sections.

III Landmark Representation

When a robot is moving through an environment, it senses and returns data readings depending on the sensors attached to it. A trace can generate a large amount of raw data that hinders the learning process. TOPSY uses a natural landmark identification process [4] to produce a smaller set of more meaningful information. From the laser sensor readings, the process identifies three kinds of landmarks: (1) discontinuities, defined as a meaningful variation in the measured distance of two consecutive readings of the laser, as shown in Figure 2, (2) corners and (3) walls, identified using a fast local Hough transform.

A natural landmark is represented by a tuple of four attributes: (D_L, θ_L, A, T). D_L and θ_L are the distance from the landmark to the robot and the orientation of the landmark relative to the robot respectively. T is the type of the landmark; *l* for left discontinuity, *r* for right discontinuity, *c* for corner and *w* for walls. A is a distinctive attribute and its value depends on the type of the landmark; for discontinuities A is depth and for walls A is its length.

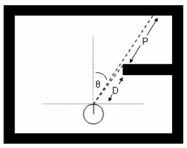


Figure 2: An example of a discontinuity.

IV Learning Navigation TOPs

The objective of our learning process is to provide a mobile robot with abilities to move through office environments and to accomplish goals. In order to learn how to move in a dynamic environment to a particular goal location, the robot has to be able to avoid, possibly dynamic, objects and to orient itself towards the goal. This paper shows how such simple skills can be learned as TOPs from robot traces and used to solve more complex tasks when combined together.

The learning process was developed under the Player/Stage robot platform [9]. The robot model was a Pioneer 2 with a laser sensor and a 16 sonar ring. To generate the behavioural clones, a human operator steered the simulated robot with a joystick. The operator executed the actions according to the sub-task being learned. While the robot was guided, the landmark identification process turned the raw data into corners, walls and discontinuities information, and a log file was recorded. Some instances of a general trace are shown in Table 1 where X_R and Y_R are the robot coordinates relative to the map of the environment; θ_R is the robot's orientation within a range from 0 to 360 degrees. D_L ,θ_L, A and T are the four attributes describing the landmark. RO was generated using the sonar ring and it indicates if there is an obstacle at the rear of the robot. The class attribute $Action$ is obtained from the robot's odometry: *forward* represents a displacement speed of at least 0.3 m/s, and *right*/*left* represents a 5 deg/s turnspeed. A raw data trace has 191 attributes: 180 from the laser sensor, 8 from the sonar ring, 3 from odometry and the class attribute. After the transformation, the high-level trace has 4 attributes for the robot and 4 attributes per landmark plus the class. This information is transformed into a small set of predicates, such as: $robot(PosX, PosY, ThetaR)$, $landmark(Distance, ThetaL, Att, Type)$, etc.

TOPSY is also able to learn new relations that can be incorporated into the background knowledge and

X_R	Y_R	θ_R	D_L	θ_L	A	T	RO	$Action$
-6.35	2.80	0.00	5.54	-63.27	0.00	c	n	fwd
-6.35	2.80	0.00	6.29	-84.06	1.48	w	n	fwd
-4.10	2.80	1.00	4.95	-90.00	0.43	w	n	right
-4.10	2.80	1.00	1.10	83.25	0.25	w	n	right
-7.13	1.61	168.00	5.07	74.34	0.00	c	n	left
-7.13	1.61	168.00	5.53	83.25	0.39	w	n	left

Table 1: General trace

used in the definition of new TOPs. The process is essentially the same, however, the examples are given from static positions of the robot rather than dynamic traces, and consequently there is no related action. TOPSY learns two types of predicates with the following format:

- $p(State, Zone) \leftarrow Conditions$ to represent new predicates, where p is the name of the predicate, $State$ is the set of predicates obtained from the current state, and $Conditions$ is a set of predicates that have to be satisfied for p to be true. $Zone$ indicates the area that the robot has to identify (e.g. "safe-turn").
- $t(State, Action) \leftarrow Conditions$ to represent TOPs, where t is the name of the TOP and $Action$ is the action performed by the robot when $Conditions$ are true.

We will describe the concept: "safe-turn", and two TOPs: obstacle avoidance and orientation, learned by TOPSY in order to combine them and accomplish the task Goto.

Safe-turn concept. It is defined as a 0.5 m free band surrounding the robot. Recognizing this area helps the robot to execute the orient action without conflicting with the obstacle avoidance task. Suppose that the robot is moving straightforward to the goal and suddenly an obstacle appears between them. The obstacle avoidance TOP will turn the robot until it finds free space in the front zone. If the robot executes the orientation action inmediately after, the robot fall into a "avoid-obtacle - orient" loop. This predicate helps to define a zone where there is no longer a close obstacle and consequently is safe to turn towards the goal. We took "snapshots" of safe and unsafe zones. The same traces of the orient TOP were used (see below). In addition, we recorded 7 traces of the unsafe zones.

Obstacle avoidance TOP. In this task, the robot has to wander around an office-like environment without collisions. The human operator avoided fixed obstacles located in different positions and distances from the robot. Each identified landmark

```
avoid(State,left) ←
    obstacle(State,distlt(0.48), anglt(-16.59)).
avoid(State,right) ←
    obstacle(State,distlt(0.49), anglt(90.0)).
avoid(State,forward).
```

Table 2: Obstacle avoidance rules

is considered an obstacle. The logged trace had 469 instances distributed as follows: *right* (44), *left* (32) and *forward* (393). The human operator changed the robot's direction when it was roughly 0.5 m from an obstacle. The background knowledge includes the definition for *less-than*. The output rules are shown in Table 2.

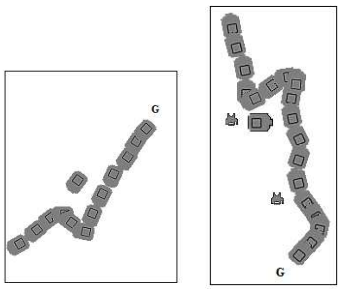
Orientation to a point TOP. Delivering documents, mail and coffee are common office tasks that can be assigned to a robot. The robot must know how to orient itself toward its target position and when to apply the orientation action. Given a target point, the robot has to turn until it is oriented toward the goal. The robot is able to execute this action only if it is located in a "safe-turn" zone. We recorded 55 traces covering the four quadrants of the space. The resultant file after merging the log files had 2982 instances: 1194 instances for left, 993 for right and 895 for nothing.

By recording the traces and induce the TOPs with Aleph, we are showing the robot what to do, instead of how to do. Consequently, the programming labor becomes easier. Once the TOPs are learned, we can combine them to accomplish a complex task.

Combining sub-tasks: Goto TOP. Given a goal position, the robot has to reach it and in the process has to be able to avoid dynamic obstacles. At this stage the user has to define a predicate to recognize a desirable goal and has to order the TOPs learned by TOPSY. As future work, we plan to automatize this following traces and recognizing the application of intermediate TOPs. Given a predicate to recognize a goal and the previously learned TOPs, we can build a TOP that corresponds to the "go-to-goal" task. It is interpreted as follows:

```
        in_goal(State,nil)
otherwise, try
        avoid_obstacle(State,Action)
otherwise, try
        orient(State,Zone,OrientAction)
```

The first rule corresponds to the main goal, whose action is nil. The obstacle avoidance TOP has the highest priority. If the path is clear, the robot is allowed to select any action from the other TOPs, otherwise, the obstacle avoidance action will be executed. If there is no obstacle, the robot verifies if it



(a) Dynamic obstacle (b) Fixed obstacles

Figure 3: The goto task

has reached the goal, if it is true, it stops. While the robot is not in the goal he has to identify if he is located in an "safe-turn" zone, and get oriented if it is true, otherwise, it indicates that there are obstacles surrounding the robot and he has to avoid or move forward.

V Experiments and Results

The **obstacle avoidance** TOP has been tested in both simulated and real robots. Markovio, a Nomad Scout robot had to amble in a cluttered laboratory. Despite the learning was made with simulated data and the robot model was a Pioneer 2, Markovio wandered safely for about one hour.

The **goto** task was tested in different situations, goal positions, fixed obstacles, user-movable obstacles and dynamic obstacles (other clones in the environment). In Figure 3(a) is shown how the robot avoids another cloned robot, get oriented again toward the goal and reach it successfully. Figure 3(b) shows the same task but the robot had to avoid two fixed obstacles. The main advantages observed are: (i) The approach facilitates the programming of a robot and it is suitable to deal with different goals in dynamic environments, (ii) The output are rules that can be re-used in different tasks, (iii) the robot can reach a goal combining learned sub-tasks, (iv) if there are changes in the environment, the robot doesn't have to learn again.

VI Conclusions and Future Work

This paper presents a learning system for navigation TOPs for mobile robots in office environments. It is based on a teleo-reactive framework, suitable to manage tasks with multiple goals. We use a relational representation that allows to deal with the problem of the huge amount of data generated by the robot's sensors. Besides, the output rules obtained by the ILP system can be re-used in another tasks.

As part of our future work, we are working on extending the approach by adding zones like: corridor-zone, closed-room zone, door-zone and associate an action depending of the goal location. It will help the robot to avoid falling into traps, e.g. if the goal is in the opposite direction of a closed room and the robot should not enter to it.

VII Acknowledgements

The first author is grateful to CONACyT for its support, grant #203873.

VIII References

[1] Benson, S., and Nilsson, N. J. Reacting, planning, and learning in an autonomous agent. *Machine Intelligence 14* (1995), 29–62.

[2] Bratko, I. Modelling operator's skill by machine learning. *22nd Int. Conf. Information Technology Interfaces* (June 2000).

[3] Broda, K., and J.Hogger, C. Designing and simulating individual teleo-reactive agents. *Poster Proceedings, 27th German Conference on Artificial Intelligence, Ulm* (2004).

[4] Hernández, S., and Morales, E. Global localization of mobile robots for indoor environments using natural landmarks. *IEEE International Conference on Robotics, Automation and Mechatronics (RAM)* (2006).

[5] Konik, T., and Laird, J. E. Learning goal hierarchies from structured observations and expert annotations. *Machine Intelligence 64* (2006), 263–287.

[6] Michie, D., and Sammut, C. Behavioral clones and cognitive skill models. *Machine Intelligence 14* (1995), 395–404.

[7] Nilsson, N. Teleo-reactive programs for agent control. *Journal of Artificial Intelligence Research 1* (1994), 139–158.

[8] Srinivasan, A. The aleph 5 manual http://web.comlab.ox.ac.uk/oucl/research/areas /machlearn/ aleph.

[9] Vaughan, R., Gerkey, B., and Howard, A. On device abstractions for portable, reusable robot code. *Proceedings of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS) http://playerstage.sourceforge.net* (2003).

[10] Zelek, S., and Levine, M. D. Spott: A mobile robot control architecture for unknown or partially known environments. *AAAI Spring Symposium on Planning with Incomplete Information for Robot Problems* (1996).