Learning Navigation Teleo-operators with Behavioural Cloning

Blanca A. Vargas Govea, Eduardo Morales Manzanares, Sergio Hernández Alamilla
Department of Computer Science
blanca@inaoep.mx, emorales@inaoep.mx, sergio.alamilla@gmail.com

ABSTRACT

Programming a robot to perform tasks in dynamic environments is a complex process. Teleo-operators (TOPs) have proven to be an effective framework to achieve goals when unexpected events occur. Their definition, however, is a difficult and time-consuming process. In this paper, two learning systems are presented: (i) TOPSY, that learns TOPs from human-guided traces of simple mobile robot tasks, and (ii) FOSeq, an algorithm that extracts grammars from robot's trajectories. TOPSY transforms large sets of low-level sensor readings 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. FOSeq uses human-guided traces of more complex tasks and the previously learned TOPs to generate sequences of TOPs from which a grammar is induced. It is shown that the learned TOPs are able to solve tasks in different dynamic environments.

I INTRODUCTION

When people go to a new place, e.g., a conference site, they normally ask for directions of places of interest, like the registration desk or a toilet, and are given general directions, like "at the end of the aisle to your right" or possibly "in room 203". People have to navigate without collisions in an unknown and dynamic environment to a particular destination point through well known natural marks like walls and doors and expected dynamic conditions, like walking people. Imagine you want your robot to learn how to perform a similar skill. You place your robot in an unknown environment and you want it to navigate to a particular point, like a charging station, but the robot is only given the general direction of its destination point. The robot has to learn how to perform simple skills, like obstacle avoidance and orientation towards a goal, and use them to safely go to a particular goal in a dynamic and unknown environment.

The application of machine learning techniques with expressive representation languages to domains like robotics have received little attention due to the huge amount of low level and noisy data produced by the sensors. The use of relational representations in this area are novel, and their advantages have recently being addressed, as in [3].

In this paper we present a relational approach to learn robot tasks. We are using behavioural cloning, a technique to learn skills from examples. The key idea of this method is to show the robot what to do instead of how to do a task (e.g. steering the robot avoiding obstacles), simplifying the programming effort. The examples or traces are processed by a learning algorithm that obtains a model which describes the task. In our approach the output is a set of reactive control rules known as teleo-operators (TOPs) [1]. The traces consist of low-level sensor readings that are transformed by TOPSy into a small set of high-level concepts based on natural landmarks. This transformation reduces the data to be processed by an Inductive Logic Programming algorithm. Once the basic TOPs are obtained, more complex tasks are learned with FOSeq, an algorithm that induces grammars from robot's trajectories. The contributions of this paper are: (i) we propose a novel relational approach that transforms raw data into high order rules for robot navigation (ii) we show how to learn basic TOPs, (iii) we show how to learn hierarchies of TOPs. We test the approach in different scenarios and show that the robot is able to accomplish several navigation tasks with the learned TOPs.

This paper is organized as follows. Section 2 presents an overview of TOPSY. Section 3 describes the natural landmark representation. Section 4 explains how TOPSY learns relational concepts and TOPs. Section 5 describes how to extract rules from trajectories with the FOSeq algorithm. Section 6 shows experiments and results, and finally, conclusions and future research directions are given in Section 7.

II AN OVERVIEW OF TOPSY

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

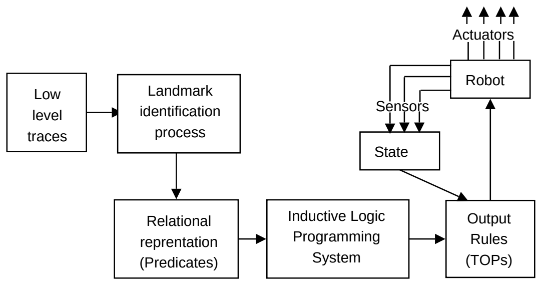

Actuators
Robot
Sensors
State
Low level traces
Landmark identification process
Relational reprentation (Predicates)
Inductive Logic Programming System
Output Rules (TOPs)

Figure 1: Block diagram

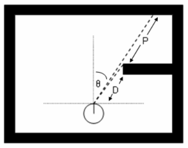


Figure 2: An example of a discontinuity.

- 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.

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.

Given a trace of low-level sensor data, TOPSY creates a trace with natural landmarks information, with the robot coordinates (X_R,Y_R) relative to the map of the environment, the robot's orientation within a range from 0 to 360 degrees (θ_R), information if there is an obstacle at the rear of the robot (RO) using the sonar ring; and the class attribute or *Action* obtained from the robot's odometry: *go_forward* represents a displacement speed of at least 0.3 m/s, and *turn_right*/*turn_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. These traces are transformed into traces of Prolog facts and reduced by eliminating duplicates with the following format:

```
task_name([robot(Xr, Yr, Thr),rear(V), goal(Xg, Yg, Thg),
  landmark( Dist_1,Thl_1,Att_1,Type_1),...,
  landmark( Dist_n,Thl_n, Att_n,Type_n),...], Action).
```

where the first argument is the current *State* with information of the robot position, whether there are obstacles at the rear of the robot, the goal position (if known), and all the recognized natural landmarks, and the second argument is the recognized action.

Teleo-operators (TOPs) are sets of reactive rules that sense the environment continuously and apply actions whose continuous execution eventually satisfy a goal condition.cl To learn a TOP, TOPSY needs: (i) the name of the target predicate (given by the user), (ii) a set of positive and negative[1] examples, (iii) possibly additional background knowledge, and (iv) a predicate, given by the user, to identify when a goal is satisfied. To induce the rules, we use Aleph [5] as our ILP system.

TOPSY learns a set of clauses with the following format: $top(State, Action) \leftarrow Conditions$, where top is the name of the TOP and $Action$ is the action performed by the robot when $Conditions$ are true. $Conditions$ is a set of predicates given

[1]Automatically generated from the traces.

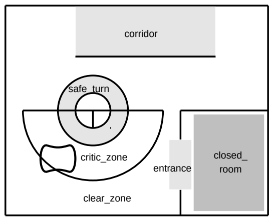


Figure 3: Environment zones

as background knowledge. The first clause is automatically generated by TOPSY with the null action and the goal predicate defined by the user as condition. There is a trade-off between the amount of user-defined knowledge and what it required from the ILP system to learn. In this paper some basic predicates are given by the user: less than or equal (*lteq/2*), greater than of equal (*gteq/2*), the selection of the closest natural landmark identified by the system (*landmark_min/4*), the current angle towards the goal (*getpose/N*), if the angle of a target is closer to the left (*closer/2*) and a predicate to identify if the robot is oriented towards the goal (*headings/3*). These six predicates are used to learn other new predicates that are used as conditions of different TOPs using the same framework, i.e., an ILP system and traces of examples. Fig. 3 shows some of these common concepts, that express mainly environment conditions necessary to perform particular actions, like *critic* zone (when obstacles are dangerously close to the robot), *safe* zone (when there are no close obstacles in front of the robot), *safe_turn* (when there are not obstacles around the robot), etc.

TOPSY learned the following predicates that are used as background knowledge to learn TOPs: (i) *frontzone*(*State*,*Value*): to express if there is an obstacle in front of the robot, where *Value* can take the values of *critic* or *clear*, (ii) *orientzone*(*State*,*Value*): to express if it is safe to turn around (no obstacles in front or behind the robot), where *Value* can take the values of *safe_turn* or *non_safe*. Similarly, TOPSY is able to learn definitions for other predicates useful for other TOPS like *entrance*, *closed_room*, etc.

Navigation TOPs. TOPSY used the previously learned predicates and its original background knowledge to learn the following TOPs:

- *avoid(State,Action)*: to wander around without collisions. *Action* can take the following values: *go_forward* (in this case, it represents the null action, since it is assumed that the robot is continuously moving), *turn_left*, and *turn_right*. In this case *frontzone(State,clear)*, that was previously learned, is given as the predicate goal (condition) where there is no need to avoid obstacles, and it is included in the first clause. We show the $go\_forward$ and $turn\_left$ rules.

  avoid(State,$go\_forward$) $\leftarrow$
  frontzone(State,$clear$).
  avoid(State,$turn\_left$) $\leftarrow$
  frontzone(State,*critic*),
  landmark_min(State,Lmk,Dist,Ang),
  lteq(Ang,-1.51).

- *orient(State,Action)*: Given a target point, the robot has to turn until it is oriented towards the goal, only if it is located in a *safe_turn* zone. *Action* can take the following values: *turn_right*, *turn_left* and *nothing*.

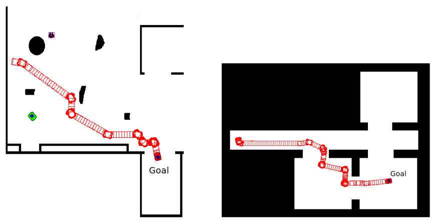
  The goal predicate, given by the user, is when the robot is oriented towards the goal expressed as a combination of *get_pose* and *headings*, and represents the conditions for the first clause. The other clauses use *orientzone* that was previously learned.

  orient(State, *nil*) $\leftarrow$
  get_pose(State, Thr, Thg),
  headings(Thr, Thg, *equal*).
  orient(State, $turn\_left$) $\leftarrow$
  orientzone(State, $safe\_turn$),
  get_pose(State, Thr, Thg),
  headings(Thr, Thg, *different*),
  closer(Thg, Thr).

V FOSEQ: COMBINING SUB-TASKS

In order to learn how to combine TOPs to perform more complex tasks, previously learned TOPs that apply to states in traces are identified, returning high-level traces of applicable TOPs.

The goal is to learn a grammar with TOPs able to reproduce a set of traces. With this purpose, FOSeq (First Order learning from Sequences), an algorithm to learn grammars from sequences is introduced. Finally, the generation of new clauses are produced as follows: each non-terminal symbol is replaced with the head predicate given by the user (e.g. goto(State,Action)) and the final set of predicates is selected according to the frequency of appearance, where clauses with less than 4 occurrences are discarded. The last step replaces constants appearing in clauses in the same order of different traces but with different values with variables. In our experiments, a reimplementation of SEQUITUR [2] was originally considered for learning the grammars, however: (i) it handles only discrete symbols, (ii) it is based on digrams (sets of two consecutive symbols), consequently, the body



(a) Several obstacles (b) Goto in a corridor

Figure 4: Goto in different scenarios

of the learned rules is limited to pairs of literals or repetitions of pairs, (iii) the uniqueness constraint is restrictive because a trace can include an important action (or a combination of them) that appears only once.

The goto TOP was learned with FOSeq and applying the simple generalization as described above: (i) between the rules of the grammar, and (ii) between sequences of the remaining trajectories.

VI Experiments and Results

The learned TOPs were tested using the Player/Stage simulator with a Pioneer 2 robot model equipped with a laser scan sensor and a ring of sonars. It is shown how the learned TOPs can be used in navigation tasks with different environments. The TOPs were evaluated by its effectiveness, measuring how well the task is completed: percentage of tasks successfully completed, percentage of obstacles successfully avoided, and number of operator interventions (e.g., if the robot enters in a loop).

Wander. Each experiment lasted 30 minutes or less if the robot entered in a cyclic path. The experiment succeeds if the robot does not collide. We run 18 experiments in four different environments. The robot completed successfully 88.9% of the tasks and it avoided 97.6% of the obstacles that it reached.

Orient. In this task the robot has to orient itself to a given point only if it is located in a safe zone, otherwise, the robot has to apply the avoid TOP and wander until it reaches a safe zone. We performed 20 experiments, which succeeded if the robot selected the appropriate zone and oriented itself towards the goal. It succeeded in 90% of the given tasks but it did not recognized two of the perceived zones.

Goto. In these experiments, the *goto* hierarchical TOP, learned by FOSeq was evaluated. Several scenarios with different obstacles' sizes and shapes were used. A task is successful if the robot reaches the goal. The robot successfully completed 82.1% of the tasks. Examples are given in Fig. 4.

VI Conclusions and Future Work

This paper introduced a two phase learning process to learn TOPs for mobile robots in office environments: (i) learning of basic TOPs (TOPSY), and (ii) learning of complex TOPs (FOSeq). In the first phase, TOPSY uses human-guided traces and a natural landmark identification process to reduce the information from the sensors into a small set of ground predicates suitable for an ILP system. A small set of background knowledge is given to the system from which other predicates are learned and used in the induction of simple TOPs. In the second phase, more complex traces are given to induce a grammar based on TOPs from which a hierarchical TOP is constructed. The learned TOPs were used for navigation tasks on different environments with dynamic objects with very promising results. As part of our future work, we are working on learning more concepts, like closed-room, entrance, etc., and more basic and hierarchical TOPs to solve more complex tasks, like how to avoid falling into traps, going to several goals, etc.

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] C.Nevill-Manning, and Witten, I. Identifying hierarchical structure in sequences: A linear-time algorithm. *Journal of Artificial Intelligence Research 7* (1997), 67–82.

[3] Cocora, A., Kersting, K., Plagemann, C., Burgard, W., and De Raedt, L. Learning relational navigation policies. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Beijing, China, 2006).

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