A CBR system for autonomous robot navigation

Raquel Ros, Ramon López de Màntaras, Carles Sierra, Josep Lluís Arcos
Artificial Intelligence Research Institute —IIIA
Spanish Council for Scientific Research —CSIC
08193 Bellaterra, Barcelona, Catalonia, Spain.
{ros,mantaras,sierra,arcos}@iiia.csic.es

Abstract. In this paper we propose the use of case-based reasoning techniques to improve the navigation of an autonomous robot in unknown semistructured environments. At the moment the current goal is to identify problematic situations (such as dead ends or obstacle layouts that the robot is not able to avoid) and take the proper actions in order to avoid them. As the first steps we propose a similarity function to retrieve similar past cases. We integrate a CBR agent into an existing multiagent navigation system in order to evaluate the performance of the CBR system. The results obtained through simulation show that the new system not only prevents the robot from getting blocked in certain situations, but also improves the performance in terms of time and distance of the path taken to reach the target.

Keywords. Case-based reasoning, Autonomous Robots, Navigation

1. Introduction

One of the main issues to solve when designing a navigation system for autonomous robots in unknown environments is the lack of a priori information about them. This lack of information impacts on many tasks like building maps, localise the robot, route planning, obstacle avoidance, etc. In this paper we report on the use of CBR techniques to reuse previous navigation experiences. These experiences can play the role of the missing a priori information needed to solve some of the above mentioned tasks.

Some researches have focussed their work on learning the environment map using probabilistic methods ([8,9]). However, we are not interested in learning the *current* map which only would be useful if the robot always navigates through the same environment. We propose to learn the different situations a robot encounters as it navigates through *several* environments. We aim at representing these situations by means of certain features that permit to characterise, for instance, dangerous situations that could lead the robot to a failure, or safe situations which may guarantee a successful performance. Completely structured environments are not too adequate for learning because it is possible to build a domain model that might be enough to determine navigation strategies. On the other hand, no learning could ever be done over an environment where objects and landmarks are randomly distributed because the probability that previous experiences could be useful is too low. Therefore, we propose to apply learning over a semi-structured world with

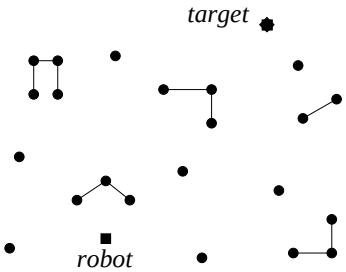

Figure 1. Semistructured environment with landmarks (circles) and linear obstacles between them forming some shapes.

landmarks and linear obstacles forming different types of shapes (Figure 1). The idea is to determine whether the current situation in the neighbourhood of a robot (landmarks and obstacles layouts) is similar to a previous situation already navigated by the robot. If this is the case, then similar actions to those executed in that previous situation might be performed now, but possibly with slight modifications, to pass through the current situation. For example, imagine a situation where the robot enters a dead end and that it is the first time this situation happens. After a while the robot will figure out that it has to go backwards only after reaching the end of the dead end. However, if this situation has been experienced, next time it encounters a similar situation, it should not enter the dead end and instead aim at finding another way to avoid the dead end. In order to model the system we will refer to two goals: global and local. The global goal is the whole navigation task, that is, reaching a given target in an unknown environment. The local goal is to successfully navigate through the local environment in the neighbourhood of the robot.

The problem now is thus how and what to learn. A first option could be just to select a number of physical layouts (landmarks and obstacles) and let the robot learn a general model of environments using some adequate machine learning technique so the model can be later used when navigating an unknown environment. An alternative is to take a lazy learning approach where for each new local environment the system is able to (*i*) identify similar previously navigated local environments, and (*ii*) exploit the decisions made in those identified local environments. That is, the two basic steps of case-based reasoning (CBR) [3].

CBR has already been used in robotics to perform different tasks. Zita and Shewchuk [2] solve a path planning problem with a system that plans a route using a city map. The global path is created using different cases from the case base. Kruusmaa [4] develops a system to choose routes in a grid-based map that are less risky to follow and lead faster to the goal based on previous experience. Ram and Santamaría [7] and Likhachev and Arkin [5] focus their work on a CBR approach to dynamically select and modify the robot's behaviours as the environment changes during navigation. Micarelli et al. [6] address indoor navigation using sonar maps. Urdiales et al. [10] present a reactive navigation system for indoor environments that learns how to avoid obstacles during navigation.

In contrast to the above previous work, in our approach we have no initial map, the map built along the navigation is a topological one, our environment is static, we use vision and finally we do not focus just on obstacle avoidance but also consider more general situations that should be avoided (e.g. dead ends).

This paper describes the initial results of a case-based reasoning system aiming to guide a robot through local environments in order to improve its navigation performance. In Section 2 we present an overview of our current navigation system. Then, in Section 3 we explain the similarity function used to identify past cases corresponding to navigation experiences. At this stage we centre our research on negative situations. That is, dangerous situations that the robot must avoid (e.g. dead ends, blocking situations, etc.). In Section 4 we report on initial experimental results. Finally, we outline some conclusions and future work.

2. Robot architecture

We introduce case-based reasoning on top of an already existing multiagent navigation system for autonomous robots [1]. The main task of this system is to navigate through an unknown environment, avoiding obstacles, and reach a target indicated by a human operator. The architecture consists of three main sub-systems:

- Pilot: is able to safely control the motors that move the robot in a given direction while avoiding obstacles using a potential field technique.
- Vision: its main tasks are to identify new landmarks in the view field of the cameras and to recognise previously identified landmarks.
- Navigation: leads the robot to the target. Its behaviour is the result of the interaction among several agents, each in charge of a concrete task:
  - *Map Manager*: is responsible of maintaining the information of the explored environment in the map.
  - *Target Tracker*: keeps the goal located with minimum imprecision and guides the robot towards it.
  - *Risk Manager*: keeps the risk of losing the target as low as possible.
  - *Rescuer*: recovers the robot from blocked situations.
  - *Communicator*: arbitrates the above agents.

To represent the environment the system computes a topological map[1] as it discovers obstacles and landmarks during the robot's navigation. A landmark is a distinguishable object which the vision system is able to identify and is used as a reference point. Obstacles are identified when the robot bumps against them. There are two types of obstacles: *point* obstacles and *linear* obstacles. The first ones are easy to avoid by slightly modifying the robot's trajectory. The Pilot can tackle them alone and they are not stored in the map. The second ones are long obstacles that block the path of the robot. In this case, the navigation system must compute an alternative route to avoid the obstacle and reach the target. Hence, they are represented in the map. To drive the robot to an alternative route the *Rescuer agent* computes a diverting target. Thus, the robot changes the current route and reaches the target following an alternative path.

In the next section we describe a new agent, *CBR agent*, that complements the previous ones in the navigation subsystem in order to improve the overall navigation behaviour. This agent will detect dangerous situations and ask the *Rescuer* to set a divert-

[1] That is, a partition of the environment into a set of triangular regions determined by three non collinear landmarks [1].

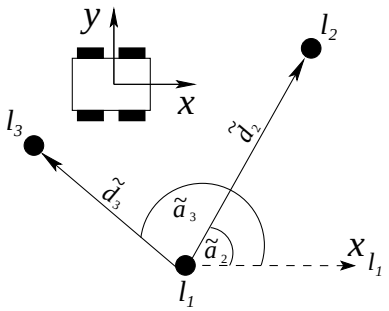

Figure 2. A $case$ with landmarks l_1, l_2, l_3 and the measures computed from landmark l_1 to the rest of landmarks.

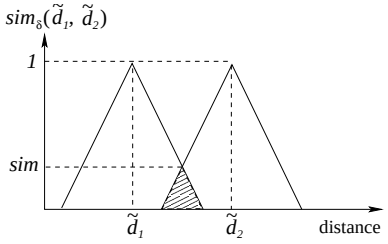

Figure 3. Similarity between two fuzzy distances.

ing target to avoid this identified dangerous situation. The integration between the CBR agent and the navigation system is described in section 3.3.

3. The CBR agent

The first fundamental decision to make when designing a CBR system is to determine what to represent in a case. In our setting the case should describe the local environment near a robot. A *case* will thus contain information about the location of landmarks and obstacles in the neighbourhood of the robot as well as distance and angular relations between them. Formally, a *case* is a particular type of graph described as:

$$case = \langle L, O, \delta, \alpha, a \rangle$$

where:

1. L: set of graph nodes representing landmark identifiers noted as l_i. $|L| \geq 2$
2. O: set of obstacles $O \subseteq L \times L$.
3. δ: total labelling function setting the distance between two landmarks, that is $\delta : L \times L \rightarrow \tilde{R}$.
4. α: total labelling function setting the angle of the line between two landmarks, l_i and l_j, and the reference axis x_{l_i} (the line passing through landmark l_i with the same direction as the x axis of the robot). That is, $\alpha : L \times L \rightarrow \tilde{R}$.
5. a represents the taken action.

Where $\tilde{R}$ is the set of triangular fuzzy numbers defined over R (see Figure 3). In Figure 2 a graphical example of a case with three landmarks is given.

3.1. Landmark Similarity Function

Given a current local environment and a memorised case, $A = \langle L_A, O_A, \delta_A, \alpha_A, a_A \rangle$ and $B = \langle L_B, O_B, \delta_B, \alpha_B, a_A \rangle$, we need a measure indicating how similar they are. This measure is reduced to finding the similarity between two graphs whose nodes are landmarks. That is, the degree of similarity between two cases is to be based on the mapping between the nodes of the two graphs. We select two landmarks (one for each

case) as the reference landmarks to compute distances and angles to the remaining ones. We note these reference landmarks as $\bar{l}_A$ and $\bar{l}_B$. Once the reference points are selected we consider that the degree of matching between any two landmarks, l_A and l_B, depends on the distance to their reference landmarks and the angle between the line connecting the landmark to the case's reference landmark and their respective reference axes, $x_{\bar{l}_A}$ and $x_{\bar{l}_B}$. Thus, we have:

$$sim(l_A, l_B) = \frac{sim_\delta(\delta(\bar{l}_A, l_A), \delta(\bar{l}_B, l_B)) + sim_\alpha(\alpha(\bar{l}_A, l_A), \alpha(\bar{l}_B, l_B))}{2}$$

The function evaluates the degree of similarity in the interval $[0, 1]$, 0 meaning no similarity and 1 meaning perfect match.

The sim_δ and sim_α functions evaluate the degree of similarity of both distances and angles. Since there is unavoidable imprecision in the measuring of distances and angles, we use triangular fuzzy numbers in the representation of these measures. Then, the similarity between distances and angles becomes the similarity between fuzzy numbers. We define the degree of similarity between two fuzzy distances, $\tilde{d}_1$ and $\tilde{d}_2$, as the maximum value of the intersection of their triangular fuzzy numbers as Figure 3 shows. The similarity between two fuzzy angles, $\tilde{a}_1$ and $\tilde{a}_2$, is computed similarly:

$$sim_\delta(\tilde{d}_1, \tilde{d}_2) = max(min(\tilde{d}_1, \tilde{d}_2))$$
$$sim_\alpha(\tilde{a}_1, \tilde{a}_2) = max(min(\tilde{a}_1, \tilde{a}_2))$$

3.2. Similarity Between Cases

Given two *cases*, A and B, with the same number of landmarks, $L_A = \{l_1, l_2, ..., l_n\}$ and $L_B = \{l'_1, l'_2, ..., l'_n\}$, we aim at obtaining the best possible match, that is, the matching between landmarks that maximises the similarity function. We therefore propose a Branch&Bound algorithm that, given the reference points $\bar{l}_A, \bar{l}_B$, finds the optimal match between landmarks. In order to apply the algorithm we need to define a heuristic function to estimate the bound and to set the constraints that restrict the configurations in the nodes of the B&B algorithm (in our case just one):

- heuristic: the maximum similarity of all possible matches will always be larger than the actual similarity of a concrete mapping between the nodes of the two cases.

$$h(\bar{l}_A, \bar{l}_B, A, B) = \sum_{l_i \in L_A, l_i \neq \bar{l}_A} \max_{l'_j \in L_B, l'_j \neq \bar{l}_B} (sim(l_i, l'_j))$$

- constraint: if there is an obstacle between l_i and l_j, and l_i matches l'_k, and l_j matches l'_m, then there must be an obstacle between l'_k and l'_m (Figure 4).

3.3. Integration into the navigation system

We define the current problem-case as the set of landmarks and obstacles in the neighbourhood in front of the robot (Figure 5). The CBR system compares this problem-case

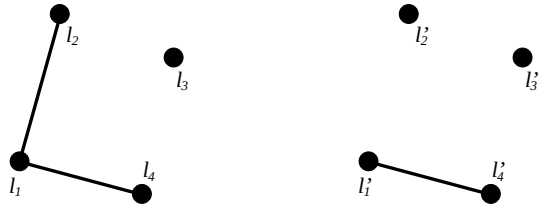

Figure 4. There is an obstacle between landmarks l_1 and l_2. Thus, the match $l_1 - l_1'$ and $l_2 - l_2'$ is not possible as there is no obstacle between l_1' and l_2'.

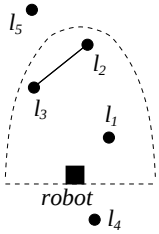

Figure 5. Landmarks l_1, l_2 and l_3 are in the region used to retrieve cases.

Scenario 1	Time	Distance
No CBR	162.36	802.13
CBR	147.87	648.13
Scenario 2	**Time**	**Distance**
No CBR	-	-
CBR	205.03	921.51

Figure 6. Average time (seconds) and distance (cm) of the path in both scenarios.

with the cases in the case base. The system retrieves those cases whose similarity is greater than a minimum similarity threshold, *simThr*. From those over the threshold (if any) the most similar is selected and its associated action is performed. In the current implementation the action is just sending a message to the *Rescuer agent* of the navigation system to compute a diverting target so that the robot avoids the current problematic situation. Once the robot gets to the diverting target, the original target is set again and the robot continues navigating through the environment as usual. During the avoiding situation phase (beginning when a diverting target is set and finishing when the robot reaches it), the CBR is inhibited to refrain from sending the same message repeatedly to the *Rescuer agent*.

4. Experiments

This section reports on initial experiments designed to compare the performance of the robot with and without the CBR system. The objective is to verify that the use of CBR would improve the navigation behaviour by decreasing the running time and distance travelled to reach targets. We have used a robot simulator to perform the experiments.

We designed two different scenarios to run the experiments and to compare the results obtained. A trial begins with the robot positioned at the initial point (point A) and finishes when the robot reaches the target (represented by a star). If the robot does not reach the target before a given time, the trial also finishes and it is considered as a failure. For each scenario, 100 trials were performed using the CBR system and without using it. Figure 6 shows the average of the results obtained.

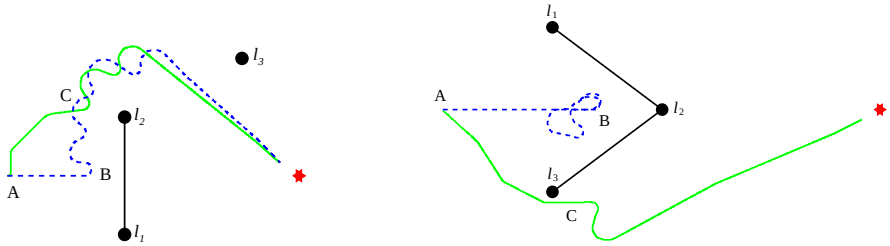

Figure 7. Path without the CBR system (dashed line), and path with the CBR system (solid line). On the left, scenario 1, and on the right, scenario 2. The target is represented by a '*'.

As we can see in the first scenario (Figure 7, left), the CBR system improved both time and distance measures required to reach the target. Without using the CBR system the robot goes straight ahead until point B where the *pilot agent* begins avoiding the linear obstacle between l_1 and l_2. Once the obstacle is passed, the robot heads towards the target. On the other hand, using the CBR system, at point A the robot retrieves a previous case formed by the landmarks l_1, l_2 and the linear obstacle. Thus, a message is sent to the *Rescuer agent* to compute a diverting target in order to avoid the current situation. The landmark l_2 is set as the diverting target. Once the robot gets there (point C), the initial target is reset and the robot continues navigating as usual. The *pilot* avoids landmark l_2 and finally the robot reaches the target. Comparing both experiments, we can see that the CBR system deviates the former trajectory preventing the robot from reaching the linear obstacle. As the results show, when using the CBR system the average time spent to complete the trial is reduced by almost a 10% with respect to the time employed by the original navigation system. Moreover, the distance is also reduced by a 20%. The optimal distance to reach the target avoiding the obstacle is 539.82 cm and the optimal time is 107 sec. (as the robot's average velocity is 5 cm/sec).

The second scenario (Figure 7 right) consists of a more difficult layout of linear obstacles. Actually, the current system (without CBR) is not able to finish any trial. As mentioned before, the pilot system of the robot uses a potential field technique to avoid obstacles. Due to the layout of the obstacles, the computed forces of the linear obstacles lead the robot from side to side in an infinite loop (point B). Thus, the robot stays there until the timeout is reached. Using the CBR system instead, at the initial point A a case formed by the landmarks l_1, l_2, l_3 and both linear obstacles between them is retrieved. Then a diverting target is computed (l_3) leading the robot to point C. Once it gets there, the *pilot* prevents the robot from getting closer to the linear obstacle and eventually the robots ends the trial. In this case, the optimal distance to the target is 868.46 cm and 173.69 sec., the optimal time.

5. Conclusion and future work

In this paper we have proposed a system able to distinguish problematic situations a robot encounters as it navigates through an unknown environment. To this end, we have used a case-based reasoning approach as it covers the main features expected for our system: identification, reuse and learning of cases. As the first steps we have designed the similarity function to retrieve, if any, the most similar cases from the case base. We

have integrated our system in the current navigation system to evaluate its performance with simple experiments. As we could observe, the CBR system greatly improves the performance of the robot in terms of time and distance of the path used to reach the target. With these positive results we are ready now to continue completing the system, with the adaptation and learning modules of the CBR system. We also plan to study the actions the CBR system should perform to guide the robot avoiding the situations it encounters, based on the decisions made in previous situations. Besides, more complex scenarios must be tested to evaluate the performance of the CBR system. Finally, it will be interesting to include positive cases representing situations that ensure safe paths for the navigation of the robot.

Acknowledgements

This research has been partially supported by the Spanish Ministery of Education and Science project DPI2003-05193-C02-02. Raquel Ros holds a scholarship from the Generalitat de Catalunya Government.

References

[1] Dídac Busquets. *A Multiagent Approach to Qualitative Navigation in Robotics*. PhD thesis, UPC, 2003.

[2] Karen Zita Haigh and Jonathan Richard Shewchuk. Geometric similarity metrics for case-based reasoning. In *In Case-Based Reasoning: Working Notes from the AAAI-94 Workshop*, pages 182–187, 1994.

[3] Janet L. Kolodner, editor. *Case-Based Reasoning*. Morgan Kaufmann, 1993.

[4] Maarja Kruusmaa. Global navigation in dynamic environments using case-based reasoning. *Auton. Robots*, 14(1):71–91, 2003.

[5] Maxim Likhachev and Ronald C. Arkin. Spatio-temporal case-based reasoning for behavioral selection. In *ICRA*, pages 1627–1634, 2001.

[6] A. Micarelli, A. Neri, S. Panzieri, and G. Sansonetti. A case-based approach to indoor navigation using sonar maps. In *6th Int. IFAC Symp. On Robot Control SYROCO 2000*, pages 345–350, 2000.

[7] A. Ram and J. C. Santamaria. Continuous case-based reasoning. In *Proc. of the 1993 AAAI Workshop on Case-Based Reasoning*, pages 86–93, Washington, DC, 1993.

[8] R. Simmons and S. Koenig. Probabilistic robot navigation in partially observable environments. In *International Joint Conference on Artifi cial Intelligence*, 1995.

[9] Sebastian Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artif. Intell.*, 99(1):21–71, 1998.

[10] C. Urdiales, J. Vázquez-Salceda, E.J. Perez, M. Sànchez-Marrè, and F. Sandoval. A cbr based pure reactive layer for autonomous robot navigation. In *Proceedings of the 7th IASTED International Conference on Artifi cial Intelligence and Soft Computing*, pages 99–104, 2003.