Adaptive Path Planner for Highly Dynamic Environments

Jacky Baltes and Nicholas Hildreth

Centre for Image Technology and Robotics
University of Auckland,Auckland
New Zealand
j.baltes@auckland.ac.nz
http://www.citr.auckland.ac.nz/~jacky

Abstract. This paper describes adaptive path planning, a novel approach to path planning for car-like mobile robots. Instead of creating a new plan from scratch, whenever changes in the environment invalidate the current plan, the adaptive path planner attempts to adapt the old plan to the new situation. The paper proposes an efficient representation for path that is easily amendable to adaptation. Associated with the path planner is a set of repair strategies. These repair strategies are local methods to fix a plan to compensate for object movement in the domain. The repair strategies are specific and have a high probability of being able to fix a plan. An empirical evaluation shows that adaptive path planning is suitable to highly dynamic domains, such as RoboCup. Adaptive path planning reduces the cumulative planning time by a factor of 2.7 compared to Bicchi's planner. At the same time, the quality of the plans generated by the adaptive path planner were similar to those generated by Bicchi's planner.

1 Introduction

Navigation is an important task for any mobile robot. Before a robot can affect the world in any sensible way, it must be at the right place at the right time. The navigation task can be broken down into a number of important sub tasks: localization, path planning, and plan execution.

Path planning is the problem of creating a collision free path through a set of obstacles from an initial to a goal position. A simple example is planning to go from the entrance of the CITR lab to the desk in room 305 on the third floor. Path planners can be categorized based on whether global or local information about the environment is available and whether objects are static or dynamic.

The F180 league in RoboCup, a game of soccer between autonomous robots, is a good example of a global path planning domain with dynamic obstacles. The RoboCup domain has a number of properties, which makes it interesting for real world planning. It features a high density of highly dynamic obstacles. This means that an agent has to re-plan often, since plans may be invalidated through objects moving. In fact, RoboCup features an active opponent that tries

to prevent a robot from executing its plan (e.g., a defender tries to prevent a striker from moving into a position in front of the goal) successfully. A robot designed to perform real world applications, such as search and rescue in dangerous environments will have to cope with similar environments.

A path planner in such a domain needs to be able to react quickly to changes in the environment. Furthermore, the path planner needs to be able to return at least an approximate plan (a best guess) immediately, that is it must be an *anytime* path planner. On the other hand, as more planning time is available, a better plan should be returned. For example, if a truck is barreling down on a robot, the planner must return an escape plan (run left) immediately. With more planning time, a complex plan for the robot to reach its destination can be developed.

This paper discusses a novel approach, adaptive path planning, for global path planning in dynamic domains. Although the approach is applicable to holonomic robots, the description focuses on car-like robots, since path planning is a more interesting problem for these non-holonomic robots.

The motivation for the adaptive path planner is to reuse previous planning work and to adapt a plan to a changing world rather than to re-plan from scratch. The adaptive planner has been tested on sample problems derived from the RoboCup competitions and during the actual competition at RoboCup-99. An empricial evaluation shows that the cumulative runtime of the adaptive path planner is 2.7 times faster than that of a path planner based on Bicchi's work, but optimized for the RoboCup domain.

Section 2 gives a brief introduction to related work. Section 3 describes the design and implementation of the adaptive path planner. The section focuses on the novel plan representation as well as the set of repair strategies. The results of the empirical evaluation are shown in section 4. Section 5 concludes and discusses how the desribed ideas can be applied to holonomic robots.

2 Literature Review

There have been many different approaches to path planning, both for holonomic and car-like robots. The major approaches for holonomic path planners are skeletonization methods (e.g., visibility graphs, Voroni diagrams, or road maps), cell decomposition (e.g., approximate or exact), or local approaches (e.g., potential fields, landmarks) [5, 3].

There are a number of problems with the path planning methods listed above. Firstly, these algorithms have a high computational cost. Secondly, these algorithms are not anytime path planners, since they require a large pre processing step. For example, quad tree decomposition, a popular approximate cell decomposition, has to break the domain up into cells before any planning can occur. Therefore, visibility graphs are the most popular global path planning methods.

A number of non-holonomic path planning methods are two stage approaches; a holonomic path planner is used to create a holonomic path, which is then converted into a non-holonomic path. Two stage approaches are not suitable to

anytime planning, since there is no gurantee that the holonomic plan can be converted into a non-holonomic one.

Other approaches use variations of standard holonomic path planners to create a non-holonomic path directly. An example of this approach is Bicchi's path planner, which is an extension of visibility graph algorithms [1]. The idea behind Bicchi's path planner is to add circles of minimum turn radius around all vertices of obstacles. The planner then finds a non-holonomic path by searching the connections between circles from the start to the goal. An example of Bicchi's path planner is shown in Fig. 1.

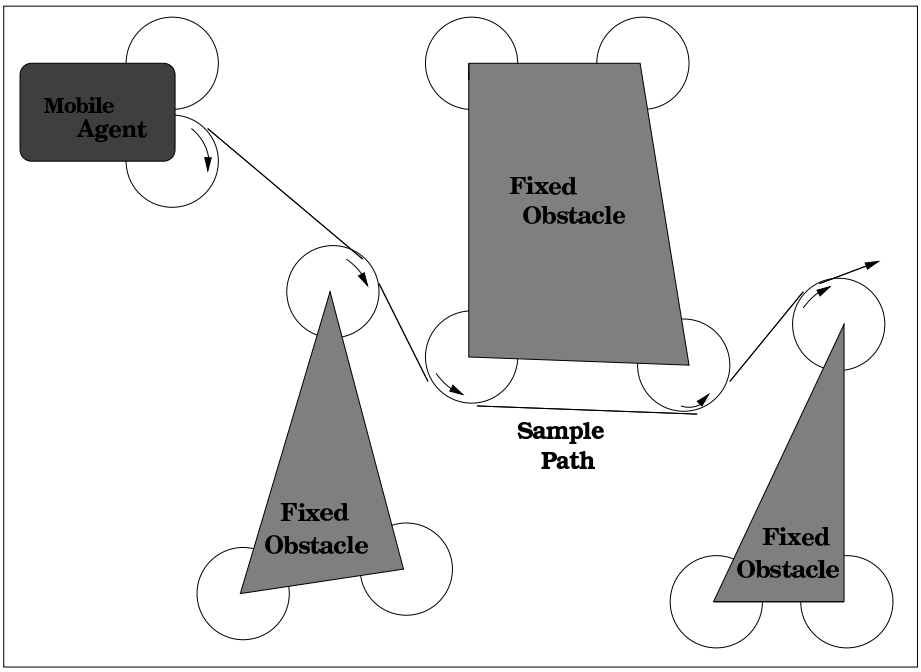

Fig. 1. Bicchi's Path Planner. A path consists of straight lines between circles located at the vertices of the obstacles.

There are a number of heuristics that can be exploited in the RoboCup domain to speed up the search step in Bicchi's path planner. For example, since all obstacles are modeled as circles, only a single circle around this obstacle is needed. Also, the search heuristic can construct the turning circles dynamically instead of creating all the connecting lines in a preprocessing step.

3 Adaptive Path Planning

Observation of a RoboCup quickly leads to the realization that most object movements from one frame to the next are rather small. Although it is true that a tiny movement can invalidate a plan or indeed make this particular planning

problem unsolvable (e.g., two opponent robots are covering the ball), in most cases, the necessary changes to the plan are minimal. So, the main motivation behind adaptive path planning is that:

- Path planning is an expensive operation, so the result of this work should be reused if possible.
- The result or output of a path planner is a partial or complete path
- Assuming that changes in the domain are small between individual planning episodes, the best plan for the current situation will be structurally similar that for the previous situation.

This motivation is similar to Hammond's case-based planning ([2]) with some important differences. Firstly, case-based planning assumes that a plan database exists with previous plans and that the most similar plan to the current situation can be found in the database. Case retrieval from the database uses commonly a similarity metric. Secondly, the database of previous plans needs to be maintained; new plans need to be added so they can be reused in the future or old plans that are not useful must be removed. So, a lot of work on case-based planning focuses on the design of suitable similarity metrics (i.e., how to find a similar case to the current one) and on database maintenance policies (should a case be added to the database or deleted).

In adaptive path planning, we assume the existence of an albeit slow static path planner that can be used to create an initial plan. Furthermore, the previous plan is the most similar one to the current situation and that, therefore, there is no need to maintain a plan database.

3.1 Path Representation

At the heart of any path planner is the plan representation. A plan consists of a sequence of path segments. Bicchi's path planner and most other planners use a representation where each path segment is either a straight line or a maximal turn to the right or left. These representations are based on a result by Reed and Shepp, that proves that any shortest path for a vehicle consists of exactly three types of segments: straight lines, full left turn, or full right turn ([4]).

However, the shortest path may not always be the optimal one, since it may lead too close to the obstacle, may include many reversals, or may result in the robot driving backwards for long stretches.

Furthermore, this representation makes it difficult to adapt a path, since the adaptations will need to be specific for a given segment type.

To simplify the adaptation, the adaptive path planner uses a uniform representation for all path segments. Each segment is an arc that contains the following information:

- start point I
- initial bearing α
- traversal direction D specifies whether the robot should travel backwards or forwards along the segment.

– length of the arc segment L
– radius of the segment R. Straight lines are represented as arcs with a large radius. There is also a minimum radius since car-like robots are limited in the radius of their turns.
– time limit to traverse the segment T. This information is used by the strategic component. If we can not reach a point in the desired time (e.g., be in place to receive a pass), it is better to recognize this fact and pursue a different course of action (e.g., play defensively), instead of reaching the goal 30 seconds after the pass was made.
– A possibly empty attachment A. An attachment is used to attach an object to a path segment, so that if the object moves, all attached path segments will move as well.

This representation, shown in Fig. 2, proved very useful, because plans in this representation can be easily adapted to compensate for movements of objects or goal locations in the domain.

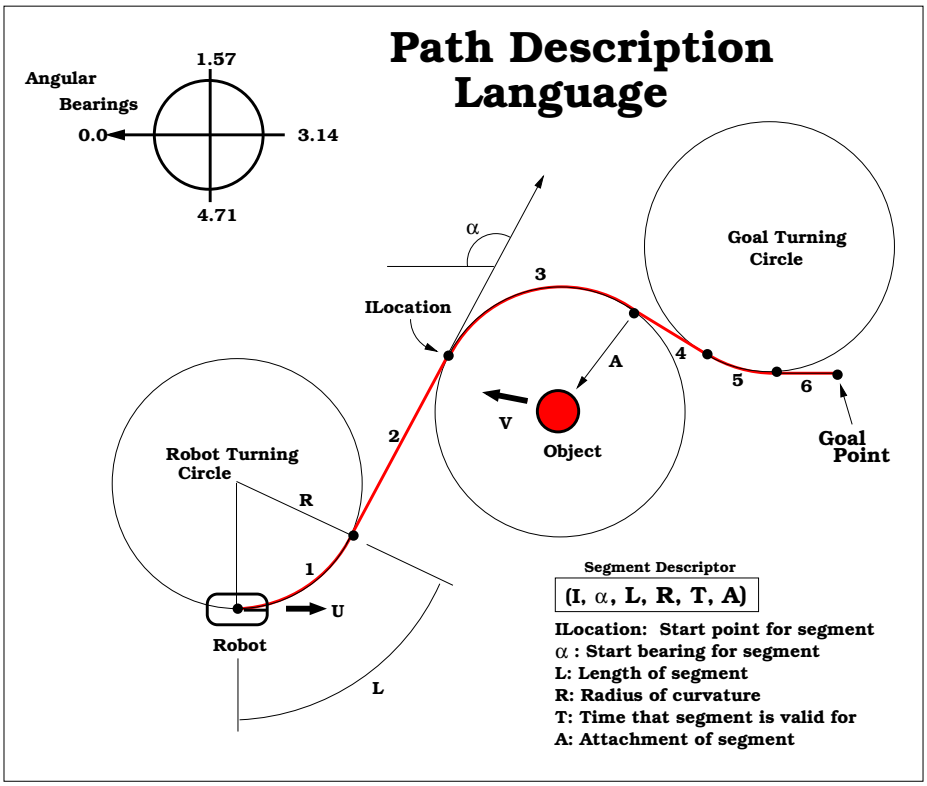

Fig. 2. Path Description Language

Objects are attached and detached from path segments dynamically. If an obstacle moves too close to a path, the path is split at the closest point to the

obstacle and a new segment is inserted which is attached to the object. As the object continues to move the attached segment will move as well. Once the object is too far from the path, the object is detached from the path segment.

3.2 Repair Strategies

Any movement in the domain results in attached path segments being moved as well, and thus may create a discontinuity, a so-called disjunction, in the path (see Fig. 3). The assumption is that repairing these disjunctions is cheaper than creating a new path from scratch.

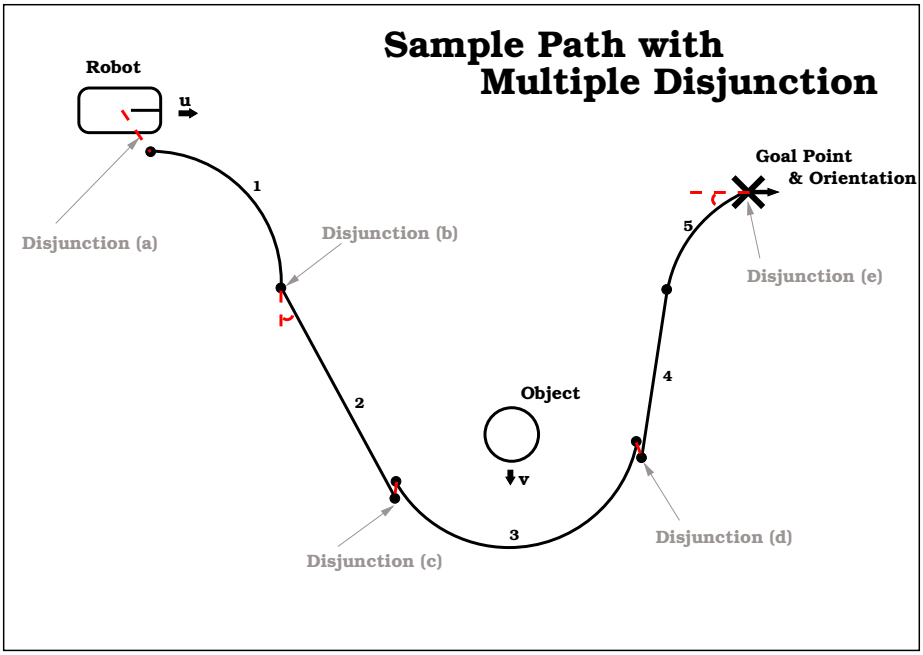

Fig. 3. An Object Movement that Results in a Disjunction

There are two types of disjunctions: distance disjunctions (break in the path) and angle disjunctions (continuous path, but discontinuity in the first order derivative).

The adaptive path planner starts with the largest disjunction and attempts to adapt the plan to fix the disjunction. In this process, new, but smaller disjunctions may be introduced into the path. The repair methods are applied using a standard A^* search algorithm that uses the complexity of the repair as a heuristic function. For example, changing the length of a segment is easier than translating a segment to a different location.

To fix these disjunctions, the adaptive path planner contains a library of path repair methods. These path repair methods are highly specific repairs that have a high chance of success and of reducing the size of the disjunction.

The following classes of repair methods are implemented in the adaptive path planner:

- **Positional Adjustment:** Changes in the start and end position and bearing of a segment. By themselves these repair strategies are not very useful, but in combination with others (e.g., shape adjustment) they can fix many plans.
- **Shape Adjustment:** These repair methods change the length or curvature of a segment. For example, a tight turn can be converted into a gentler turn.
- **Type Adjustment:** These repair strategies change the sign of the curvature, so a left turn can be converted into a straight line or a right turn. Also, the traversal direction of a circle can be swapped.
- **Segment Structure Adjustment:** These repair methods use segment insertion, segment breaks, and segment deletion to change the structure of the plan.
- **Plan Justification Adjustment:** These repair methods remove unnecessary plan segments or object attachements from the plan and thus are an optimizing post-processing step.

A shape adjusting adaptation is shown in Fig. 4. An angular disjunction is fixed by simultaneously rotating and stretching the line segment and shortening the circle.

A complete list of adaptation methods can be found in [**?**].

4 Evaluation

To evaluate the performance of the adaptive path planner, we compared the cumulative runtime and the success rate of the adaptive path planner against that of two popular non-holonomic path planners.

The evaluation was based on a set of case studies, which were derived from situations during robotic soccer competitions. Figure 5 shows a case study of a situation that was inspired by the RoboCup competitions. It shows how the adaptive path planner changes the path to compensate for a moving ball and an interfering object. The robot starts at the left side of the field. The goal location is on the right side of the field. The goal initially moves upwards until it rebounds and starts moving downward. An obstacles moves upwards in the center of the field. There are also two static obstacles in the domain.

Figure 5 shows a graph of the planning time of the adaptive path planner versus that of an optimized Bicchi planner. As can be seen, the adaptive planner uses very little time in most cases. The adaptive planner uses more time in cycle 103. That is because at this moment the obstacle intrudes into the path and a new segment attached to this object needs to be inserted. Correspondingly, the peaks at cycle 200 and 300 correspond to the path planner having to change the traversal direction around the obstacle as well as avoiding the upper obstacle.

The repair strategies had a very high success rate ($> 99\%$), since the adaptive path planner only failed in six of the 1000 planning episodes. Should the adaptive planner fail, it may or may not call the static planner in the current cycle. This

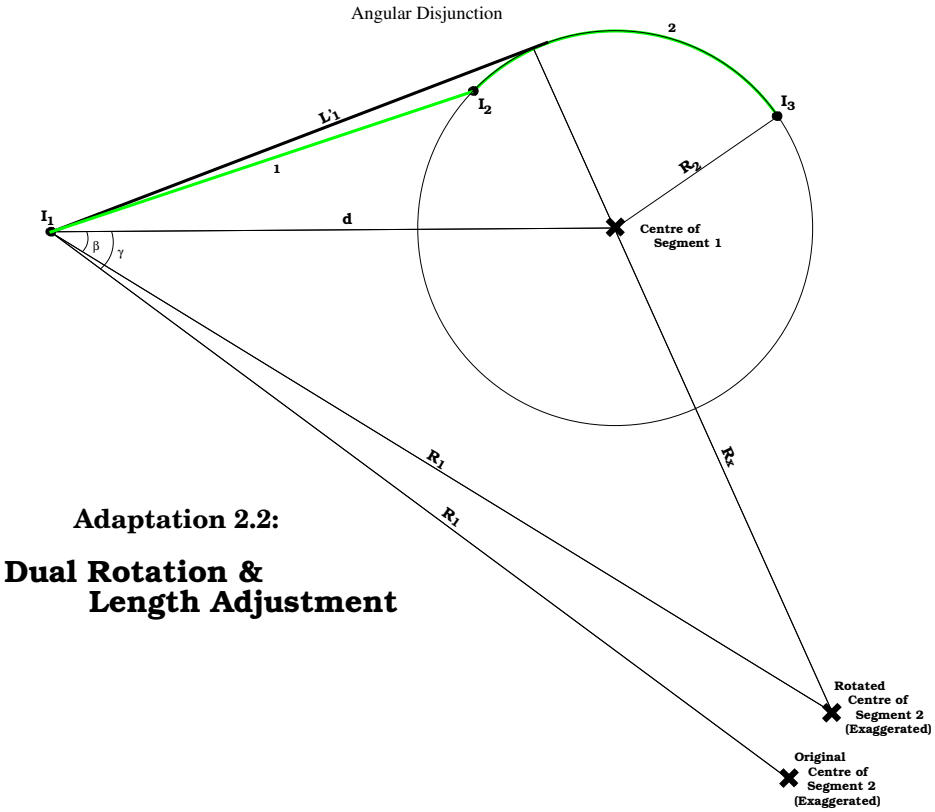

Fig. 4. A Shape Adjustment Adaptation

decision depends on the amount of computation that was done in the current cycle and the expected cost of repairing the plan.

Nevertheless, the cumulative running time for the path planner in this problem is greatly reduced. To evaluate the cumulative runtime further, we randomly created a set of real world problems similar to the one shown above. After the problem was set up using our mobile robots, the path planning activity of the robot was recorded until it reached the goal. These tests used a complex domain with 11 objects.

Table 1 shows that the adaptive path planner is significantly faster (a factor of 2.7) than the optimized Bicchi planner. Often the adaptive path planner is orders of magnitude faster. Only in situation 7, did the adaptive path planner perform worse.

The length of the plan is similar for the adaptive planner and the optimized Bicchi planner. Note that the optimized Bicchi planner uses heuristics to prefer simple path (few reversals, few segments), and not necessarily the shortest path as the original Bicchi planner.

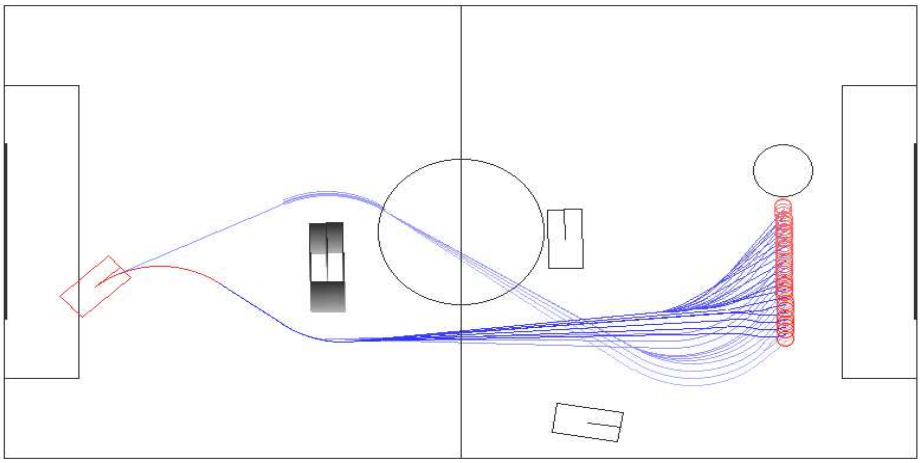

Fig. 5. Case Study Derived From Observed RoboCup Competition

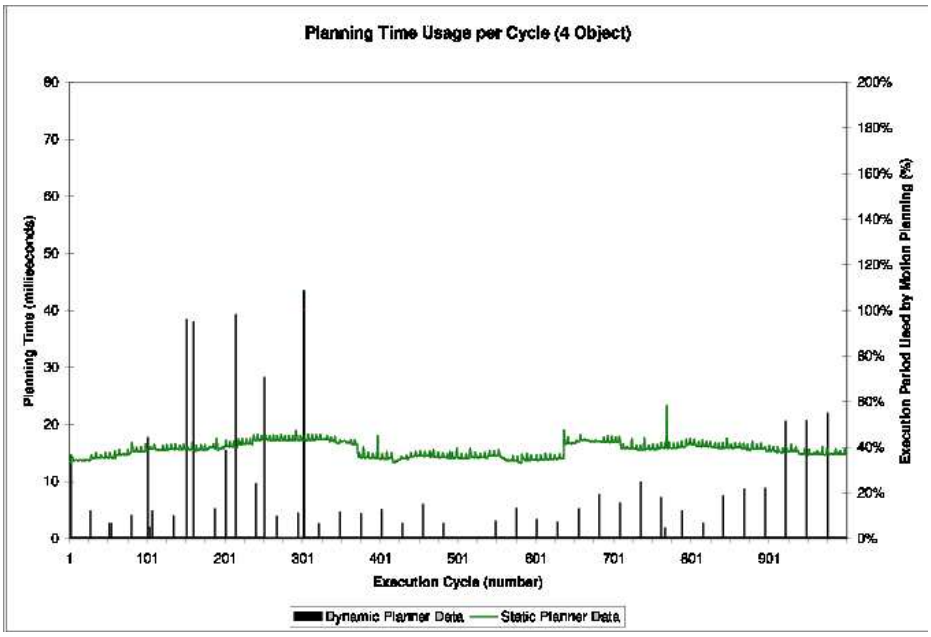

Fig. 6. A comparison of the planning times for the episode shown in Fig. 5

5 Conclusion

This paper describes a novel approach to path planning for car-like mobile robot, that attempts to reuse previous planning work as much as possible when planning for a new situation.

	Optimized Bicchi Planner			Adaptive Planner		
Sit.	Time (ms)	Outcome	Length (mm)	Time (ms)	Outcome	Length (mm)
1	$276ms$	Success	$2750mm$	$13ms$	Success	$1912mm$
2	$93ms$	Success	$1480mm$	$12ms$	Success	$1484mm$
3	$72ms$	Success	$988mm$	$2ms$	Success	$993mm$
4	$304ms$	Success	$2287mm$	$2ms$	Success	$2282mm$
5	$189ms$	Success	$2538mm$	$10ms$	Success	$2107mm$
				$23ms$	Success	$2151mm$
				$18ms$	Success	$2161mm$
6	$264ms$	Success	$2347mm$	$59ms$	Success	$1678mm$
7	$43ms$	Success	$2023mm$	$243ms$	TimeOut	$1880mm$
				$40ms$	Success	$1866mm$
8	$281ms$	Success	$1758mm$[1]	$12ms$	Success	$2758mm$
				$4ms$	Success	$2755mm$
9	$381ms$	Success	$2755mm$[1]	$242ms$	TimeOut	$3022mm$
				$8ms$	Success	$3147mm$

Table 1. Cumulative Runtime for randomly generated real world examples with 11 objects.

[1]This plan uses both forward and backward movement by the robot to complete.

A set of adaptation methods, a set of highly specific repair strategies, has been developed that have a high rate of success.

An empirical evaluation using a set planning episodes derived from real world problems shows that the adaptive planner is significantly faster than other state of the art non-holonomic planners.

The basic idea of adaptive planning is independent of the actual robot control. The only change required is the plan representation and the associated set of adaptation methods.

References

1. Antonio Bicchi, Giuseppe Casalino, and Corrado Santilli. Planning shortest bounded-curvature paths for a class of nonholomic vehicles among obstacles. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1349–1354, 1995.
2. Kristian J. Hammond. *Case Based Planning*. Academic Press Inc., 1989.
3. J.C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, 1991.
4. J.A. Reeds and R.A. Shepp. Optimal paths for a car that goes both forward and backward. *Pacific Journal of Mathematics*, 145(2), 1990.
5. Stuart Russel and Peter Norvig. *Artificial Intelligence: A Modern Approach*, chapter 20, pages 598–624. Prentice-Hall Inc., Englewood Cliffs, New Jersey 07632, 1995.