Subsumption-based Control for Mobile Robots in Dynamic Environments

Jacky Baltes
j.baltes@auckland.ac.nz

Department of Computer Science
University of Auckland
Private Bag 92019, Auckland, New Zealand

Keywords: Non-holonomic control, subsumption architecture, path planning

Abstract

This paper describes an architecture for path planning and control of car-like mobile robots. The method is based on a subsumption architecture with four individual behaviors: approach, steer, turn, and progress. The coordination of these simple behaviors results in a robust control architecture for mobile robots that performed well when compared to other control methods. The controller also results in simplifying the requirements on the path planner.

1 Introduction

The Centre for Imaging Technology and Robotics (CITR) at the University of Auckland has worked on the design of path planning, control, and task planning algorithms for the last three years. The overall goal of the project is to develop an architecture for intelligent robotic assistants and teams of assistants for humans in a variety of domains. Examples of applications for the robotic assistants are rescue operations, tour guides, space explorers, and warehouse organizers.

What all these tasks have in common is that they require robust control and path planning of the robot under real world constraints. For example, a tour guide robot must reach certain locations in a reasonable amount of time, but also not run into people or other obstacles.

Instead of using custom built robots, the "robots" used at the CITR are off-the-shelf toy cars. The toy cars were minimally modified to provide local vision and global vision platforms. The only input information is from a video camera (A 80x60 pixel CMOS colour camera for the local vision robots, a camcorder mounted on the ceiling for the global vision team). There are no other sensors, such as shaft encoders, ultrasound, or infrared, which are commonly found on other robotic platforms. In fact, for the global vision team, all processing is done off board and commands are transmitted to the car given the orignal remote control. The only difference is that instead of a human, the transmitter is connected to the parallel port of a PC. The motors and steering control are coarse, noisy, and unreliable.

As most other researchers, our initial research focused on separate methodologies for control and path planning. Given the lack of additional sensors and inaccuracies in the actuators of our robots, we quickly found that the state of the art in control for car-like robots and path planning in dynamic environments was not able to provide satisfactory results.

In the last two years, we developed several new approaches to control and path planning for car-like robots (Fuzyy-Logic [4], Reinforcement Learning [3], anytime path planning [8], and adaptive path planning [7]).

From this research, we drew two conclusions. First, a strong relationship between the controller and the path planner exists. The path planner must generate paths that the controller can follow. For example, it is of little use to generate B-spline shaped paths if the controller can follow only curves with a constant radius efficiently.

Second, a controller is useful only if the domain is static. In a dynamic domain, the controller will have to switch amongst paths too often. Therefore, a path planner should generate only an approximate path for the controller. The dynamic nature of the real world

environment, will unavoidably lead to the plan being locally modified by the controller.

From these observations, we developed the following requirements for navigation of a car-like robot in highly dynamic environments:

Anytime path planning: The time that a robot has available for path planning varies. For example, if the robot is sitting in front of an approaching truck, at plan must be returned within milliseconds. On the other hand, when the robot is waiting at an intersection, it may spend more time planning. What is needed is a path planner that can trade off plan quality versus runtime. An initial guess is available immediately, but a better plan is returned if more time is available.

Robust control: Much research on non-holonomic control is limited to controllers that have benign initial conditions (e.g., the orientation error is at most 90 degrees [1]. These systems rely on the path planner to provide a new path if the robot is outside of these initial conditions. In practice, the incompleteness of the path planners information (e.g., unknown obstacles), results in possibly large errors. Therefore, a controller must recover even from large errors. This requirement then blurs the distinction between the path planner and the controller. It is the task of the path planner to generate a plan that takes all known obstacles into consideration, whereas the controller must deal with unknown obstacles or obstacles whose trajectory has been mis-predicted. The controller needs to locally modify the plan to compensate for changes in the environment. The path planner is responsible for the global consistency of the plan.

Therefore, we focused our efforts recently on a reactive, robust architecture for path planning and control. The goal is to develop a low level controller that can approach points in sequence. Instead of a path, the path planner generates a sequence of target points.

2 Related Work

Most controllers for car-like robots try to minimize the distance and orientation error of the car. An example of such a controller is shown in Fig. 1. Point x,y is the current position of the rear axle of the car, point x_p,y_p is the closest point to the car on the path. The distance error $\tilde{y}$ is the distance between these two points. The orientation error $\tilde{\theta}$ is the difference between the orientation of the car θ and the first derivative θ_p of the path at x_p, y_p.

A problem with this representation is that it is difficult to minimize both error terms at the same time.

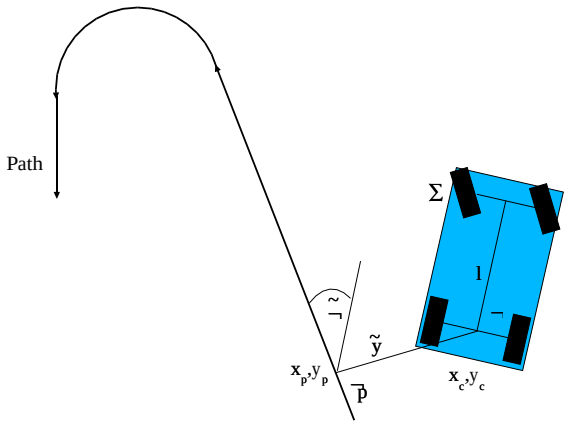

Figure 1: Representation of the Control Problem based on Balluchi

For example, the distance error is more important if the car is far away from the path, whereas the orientation error is more relevant near the path.

Instead we are using a different model, shown in Fig. 2 derived from Egerstedt [6]. The controller looks ahead along the path for a given distance d to the point x_f, y_f. The controller only tires to minimize the orientation error $\tilde{\theta}$, which is the angle between a straight line to the point x_f,,y_f and the orientation of the car. This model allows the car to approach a path gently and to look ahead along the path. In contrast to Egerstedt, we found that a simple look ahead function, which is dependent on the speed of the car, is sufficient.

Brooks in 1987 suggested so-called subsumption architectures to solve problems for mobile robots [5]. More recently, these architectures have been extended into hybrid architecture to overcome some of the short comings of the original subsumption architecture. The above model of control can easily be incorporated into a subsumption architecture. This allows the controller to be extended to deal with unpredicted events (e.g., unknown obstacles).

3 Design

This section describes the design of the subsumption architecture for the controller. The controller consists of four behaviors: approach, steer, turn, and progress. Each individual behavior is shown in the following subsections.

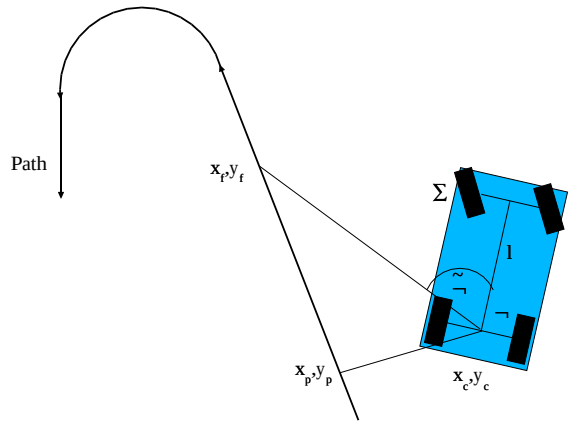

Figure 2: New Representation of the Control Problem

3.1 Approach

The first behavior is responsible for the robot following the path correctly. When the orientation error $\tilde{\theta}$ is small, then increase the speed and drive towards the control point. This behavior is not concerned with the path, but rather with the orientation and can therefore be quickly and efficiently implemented.

3.2 Steer

The second behavior is used to correct the steering when the robot is slightly of course. When the robot is facing to the right/left of the target point and the orientation error $\tilde{\theta}$ is not too big, then steer to the left/right. The amount of time of the course correction is proportional to the orientation error.

3.3 Turn

The third behavior turns the robot by roughly 90 degrees when the orientation error is large. In this case, it would take a long time to change the robots orientation. It is quicker to execute a three point turn rather than a long turn. Furthermore, continuing the path may also result in continuous large errors. An extreme example is when the target point is within one minimum turn radius of the robot. Then, if the robot is trying to follow the circle, it would simple orbit the point forever.

3.4 Progress

The last behavior is responsible for local obstacle detection and avoidance. If the robot has not made any progress towards the goal for some time (approximately 10 secs.), then it chooses a direction and steering angle randomly and drives with these parameters for a period of time (approximately 0.5 sec.). After this random behavior, the path planner is called again and a new plan developed.

3.5 Coordination of behaviors

In a subsumption architecture, many different behaviors are active at the same time and may result in conflicting actions being suggested by these behaviors. Much research in subsumption architectures is focused on the development of efficient policies for the coordination of different behaviors. Coordination of different behaviors requires an understanding of all possible interactions amongst different behaviors and is non-trivial and error prone.

A closer inspection of the subsumption based controller shows that the approach, steer, and turn behaviors are mutually exclusive, since they are all based on specific values of the orientation error. Therefore, the only possible conflict is between the progress behavior and one of the other behaviors. In this case, the current control is not working and the progress behavior should have priority over other behaviors.

Coordination of behaviors in subsumption architectures can be done through either preventing a behavior from firing by shutting of its input (inhibition) or by preventing a behavior from sending a command to the actuators (suppression). In this work, we used suppression links from the progress behavior to all other behaviors.

Figure 3 shows a graphical representation of the subsumption controller. The hierarchy of behaviors is indicated through the suppression links.

4 Evaluation

This section shows the results of evaluating the performance of the subsumption-based controller. The evaluation of the controller was done on two tasks: a race track (Aucklandianapolis) and a treasure hunt competition [2].

4.1 Performance

This subsection describes the results of evaluating the performance of the subsumption based controller against that of traditional path planners and controllers.

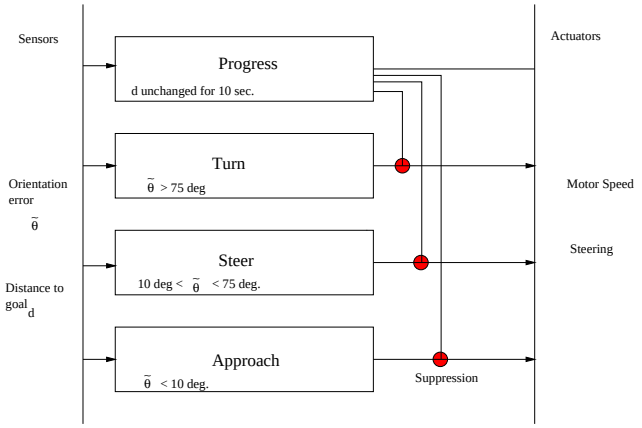

Figure 3: Subsumption Architecture for the Controller.

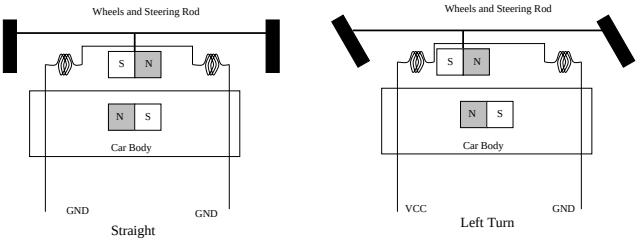

Figure 4: Steering Mechanism: An example for straight ahead (no current through the control lines) and left turn (induce magnetic field to pull the magnet to the left).

The new controller behaved performed well compared to other control algorithms that we have tested (Balluchi's controller, Egerstedt's controller, Fuzzy Logic controller, and a reinforcement learning controller). The subsumption-based controller achieved a new official track record.

This subsection shows that the subsumption based controllers performance is comparable to that of traditional control algorithms. This evaluation does not consider the additional features of the subsumption based controller such as local obstacle avoidance or robustness, which will be discussed in the next subsection.

4.2 Robustness

Robustness is an important feature of a practical autonomous robot in the real world. In particular, the robot should execute a given task, even if conditions change in new or unforeseen circumstances. Clearly, the robustness of a system to sudden change is limited. Some changes are more reasonable to expect than others. For example, one would not expect a path planner to work in worlds where obstacles materialize out of thin air, but one would expect a system to execute a task successfully, even though an obstacle moved by a few millimeters.

Therefore, evaluating the robustness of a system is non-trivial. Most evaluations change certain parameters or operating conditions. These changes are under the control of the designer of the algorithm. Since the designer is responsible for both design and evaluation, it is possible for underlying assumptions to influence the design as well as the evaluation.

This subsection provides a case study of the robustness of the described system given a real world experience. It shows that the described architecture was able to be moved to different robot platforms and cope with their idiosyncracies. Furthermore, the described changes were completely unexpected on our part. The inherent robustness of the system allowed us to solve this real world problem.

About half way through the project, we decided to start work on a local vision robot soccer team and had to buy new toy cars (Nikko Pocket Racers) for our global vision team. Instead of the proportional control, these new toy cars had a simple bang-bang control only. This did not pose a problem, since the effectiveness of the proportional control was limited and only a few steering angles were usable by the controller in practice. The bang-bang control was not achieved using a servo motor, but rather through the setup shown in Fig. 4. The figure shows two controls for straight and for a left turn. A small magnet is used to center the steering if no current is flowing through the two control lines. For a turn, a current is used to induce a magnetic field in the inductors to pull the steering to the right or the left respectively.

We were also familiar with the effect of battery charge on the velocity of the cars. For example, setting the speed of the car to 2, will result in 2.8 m/s with fresh batteries, but 0.3 m/s after a few minutes of driving. Therefore, the subsumption architecture uses standard PID controller for the velocity.

A completely novel problem occurred given the steering of the new cars. Since the magnetic field is proportional to the voltage, the steering force decreases over time, because of the discharge of the batteries. The effect is that the car has a minimum turn radius of 25 cm with fresh batteries, but only 70 cm with old ones. Most control algorithms assume a constant minimum turn radius and are not able to adjust to such drastic changes.

It was therefore interesting to see the effect that the

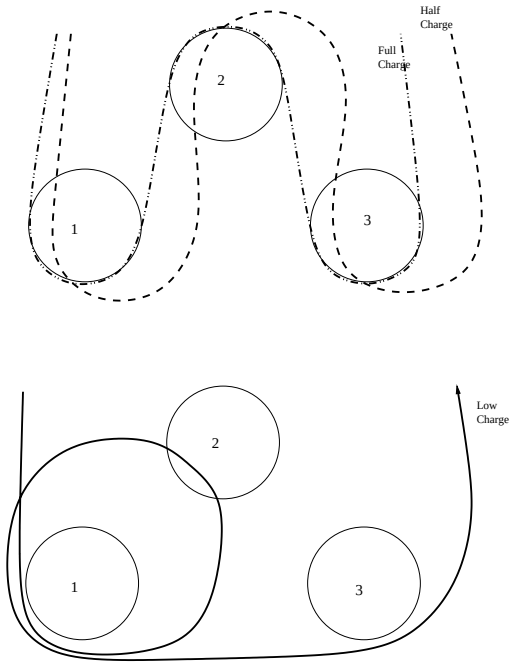

Figure 5: Track of the robot around the Aucklandianapolis race track with different battery charges. The top part qualitatively shows the track of the car with fully charged or half charged batteries. The bottom is a track with a low charge.

change in minimum turning radius has on the behavior of the controller. Figure 5 shows qualitatively the performance of the subsumption architecture on a treasure hunt. The task of the robot is to navigate around all three points in the sketch. As can be seen the controller compensates for the change in steering angle successfully. The controller is able to follow the path even with an ever increasing minimum turn radius. Once the minimum turn radius exceeds a certain limit, the controller continues to drive a circle around point 1 and then proceeds to point 3.

This subsection shows that the subsumption based controller is robust and can adapt to a variety of different conditions, even if these conditions were a surprise even to the designers of the algorithm.

5 Conclusion

The described controller has been used extensively in our experiments both at the Aucklandianapolis as well as treasure hunt competitions. It is also now used as part of the agent architecture for the All Botz robotic soccer team.

The described architecture provides a robust platform that can deal with unforeseen changes in the environment and can quickly adapt to new paths. Although not as good as the performance of other special purpose car-like robot control algorithms, the subsumption controller performs well over a wide range of conditions.

We are working on implementing additional behaviors for robotic soccer. Also, instead of using simple suppression and inhibition, we are investigating more sophisticated approaches, such as fuzzy logic and neural nets to coordinate different behaviors.

References

[1] A. Balluchi, A. Bicchi, A. Balestrino, and G. Casalino. Path tracking control for dubin's cars. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 1996.

[2] Jacky Baltes. Aucklandianapolis homepage. WWW, February 1998. http://-www.tcs.auckland.ac.nz/ jacky/teaching/courses/-415.766/aucklandianapolis/index.html.

[3] Jacky Baltes and Yuming Lin. Path-tracking control of non-holonomic car-like robots using reinforcement learning. In *Proceedings of the IJCAI Workshop on RoboCup*, Stockholm, Sweden, July 1999.

[4] Jacky Baltes and Robin Otte. A fuzzy logic controller for car-like mobile robots. In *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pages 89–94. IEEE, November 1999.

[5] Robert Brooks. A robust layered control system for a mobile robot. Technical Report 864, Massachusetts Institute of Technology, 1985.

[6] Markus Egerstedt, X. Hu, and A Stotsky. Control of a car-like robot using a dynamic model. In *Proceedings of the 1998 IEEE Conference on Robotics and Automation*, Leuven, Belgium, 1998.

[7] Nicholas Hildreth. An adaptive path planning system for car-like mobile robots. Master's thesis, University of Auckland, February 2000.

[8] Huang Li. Anytime path planning for mobile robots in highly dynamic environments. Msc., University of Auckland, Auckland, New Zealand, July 2000.