Autonomous Control of a Scale Model of a Trailer-Truck Using an Obstacle-Avoidance Path-Planning Hierarchy

Robert Woodley and Levent Acar
Department of Electrical and Computer Engineering and
Intelligent Systems Center
University of Missouri-Rolla

Abstract

A scale model of a tractor-trailer truck was developed as a testbed for control algorithms. The truck operates in autonomous or semi-autonomous modes. An on-board Pentium computer with a PC104 bus performs the computations and data collection. Various sensors and a wireless transceiver are on-board the truck. Our research focus has been in the autonomous control of vehicles using intelligent systems. For this document we have employed a multi-resolutional hierarchy to plan a path for the tractor-trailer truck. The hierarchy starts with a simple path then warps it around obstacles. The modular construction of the hierarchy will allow more intelligent agents to perform some of tasks. The current system has some limitations as to the placement of obstacles, however, it is an extremely fast algorithm and is able to handle some motion of the obstacles.

1 Introduction

In our previous work, the truck has been used as a testbed in non-linear dynamics, neuro-control, and fuzzy logic control applications [1–4]. Additional earlier work was presented in [5–7]. Figure 1 shows the scale model.

The truck is outfitted with a number of sensors and processing components. The sensors

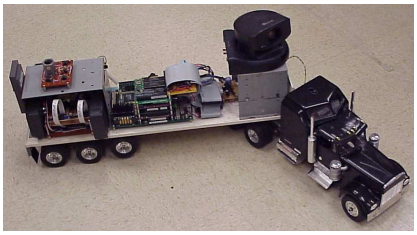

Figure 1: The scale model trailer-truck with all the components.

range from simple potentiometers to measure articulation and steering angles, to an ultrasonic, position-tracking system. The truck is also measured by an optical shaft encoder to sense speed and accelerometers to sense motion on the trailer. The main computing power is from an embedded Pentium-class computer. The computer has a PC104+ bus for compact operation. It is able to communicate with outside world via a wireless transceiver. Additionally, a micro-controller is on-board, due to space limitations, to process the signals inside the cab of the truck.

The current control scheme utilizes a multi-resolutional hierarchy to solve the problem of speed in creating the path and the ability to avoid obstacles. In a multi-resolutional hierarchy each level has a complete picture of the control system to various levels of detail. The lowest level will contain the actual control signals to the servos of the system, yet the amount

of information is limited. Each level of the hierarchy will know how to complete the command given to it by the previous level. The top level sets the goal.

The hierarchy increases the ability of path planning in two ways. First, each level has only a certain range of information to learn. Second, since the hierarchy is modular by nature, a problem at one level is likely to just require retraining of that level. This is also true for the case of changing environments.

Presently, each level of the hierarchy is a fixed algorithm rather than an intelligent agent. Much is known about the truck system and the environment we are currently using. Given this information, we developed each level's algorithm knowing what situations were likely. With the success we will show here, further intelligent agents will be developed to handle even larger classes of situations.

2 The Multi-Resolutional Hierarchy

The hierarchy we created has six levels. The top level, Level A, has the duty of determining if the target has been reached. Level B then adjusts the straight line path around any obstacle that the line crosses. Level C refines the path to eliminate any unnecessary turns from the path. Level D smooths the path so that any corner can then be navigated by the truck. Level E calculates the time steps between sample points, the velocity the truck will move, the angle profile of the trailer to follow the path, the angle profile of the cab to move the trailer, and the angle profile of the steering tires to move the cab. Level F contains a feedback loop with the truck sensors to provide the signals to the truck actuators to follow the path and angle profiles.

The action of Level A was motivated by our previous work. In our previous work we trained a neural network using training vectors developed through the use of a path planning algorithm developed by Rensselaer Polytechnic Institute (RPI) [4,8]. We showed how the approach we took improved upon work done by RPI and that it had advantages over other neural network and fuzzy logic implementations of the truck backing problem as found in [9,10]. We learned that we only had to come up with a path that was somewhat close to being viable and the control system would follow it.

Level A has as its inputs the positions of known obstacles, the current position of the truck, and if any previous path has been calculated. The current position is used to generate the straight path. If the system has already generated a path, Level A can take that as the starting path to send to the other levels. If the obstacles are all still in the same location, the path will remain unchanged. If, however, an obstacle moves to a new location that intersects the existing path, the hierarchy will refine the path to avoid this obstacle. In this way, a motion such as an oscillation of an obstacle can be incorporated into the system so that the path avoids the moving obstacle. This memory aspect will be further studied in future work.

Level B has the duty of creating straight line segments around obstacles. The obstacle locations are provided to Level B as well as the current path. The algorithm simply checks to see if a segment from the current path crosses an obstacle. If an intersection is detected, the algorithm cuts the segment that has the error. It then draws straight lines around the obstacle that will get the path closest to the other existing segments. It then connects the point at the end of the excursion around the obstacle to the next path point.

The obstacle data are provided to Level B by means of an error map. Figure 2 show an example of an error map. An obstacle and the side walls are represented by a value of one. The error then decreases to zero at a meter from the obstacle. This allows the algorithm to know if it is getting close to an obstacle.

Table 1: Level descriptions in the multi-resolutional hierarchy

Level	Action	Input	Output
A	Determine target Create initial path	Obstacle position Current position Current path	Obstacle position Current position Current path
B	Adjust path (around obstacles)	Current position Obstacle position Current path	Obstacle clearance path Obstacle position
C	Refine path (remove extra turns)	Obstacle position Obstacle clearance path	Refined path
D	Smooth path	Refined path	Smoothed path
E	Calculate movements	Smoothed path	Velocity profile Displacement profile Angle profiles
F	Control actuators (follows profiles)	Velocity profile Displacement profile Angle profiles	Steering voltage Drive voltage

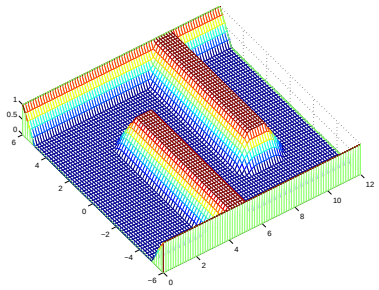

Figure 2: An example of an error-field map used by the obstacle avoidance level.

The next two levels have much simpler functions, but require more detail. Level C has the duty of removing unnecessary turns. The output of Level B can potentially create a turn that can be avoided. Level C looks at a point and checks to see if the previous and following point can be connected without intersecting an object. Level D takes the refined path and smooths the corners. The path created by Level D is stored and sent back to the top level to be used by the next iteration.

Level E has extremely detailed information about the truck. The dimensions, maximum velocity and acceleration, as well as limits on the steering are all known at this level. Level E knows what the time step size will be from one path point to the next. From the time step data and the dynamics of the truck a velocity profile can be created along the path. Then the trailer, cab ,and steering angles can be calculated for the particular path point.

In an experiment performed on the neural network controller, it was found that we could take the path generated by the neural network and warp it. When we warped it, we only stretched the x and y coordinates. We did not change the angle information. In doing so, we created situations that are physically impossible. Primarily, we had the trailer moving perpendicular to its long axis. However, when we took this warped path and entered it into the control system, the truck nearly made it to the target. During the section that was physically impossible, the truck strayed from the path, but was constantly trying to get back on track. When the truck reached the section that was not warped as much, it was almost able to get back to the path. In another experiment, it did make it to back to the path. Our conclusion from these experiments was that the truck system is linearized about the path. In this way, as long as the path stays within the region of convergence for the linear controllers on the truck the truck will eventually be able to follow the path.

Given that we only have to get close to a viable path, we have certain freedoms in calculating the path points. Between Level D and Level E we are able to produce turns that may not be possible by the truck, but if we allow enough room for the truck it will follow the path. A one meter buffer between the path and the obstacles was created for this reason.

Level F has the most detail of all the levels. Level F contains the linear controllers used on the truck. Each path position is applied to the linear controllers at a particular time. The controllers then compare the actual sensor data with desired path data to create the control signal to the actuators.

3 Simulation Results

Figure 3 shows the obstacle, the initial truck location, and the initial straight line path given to the algorithm. Without the horizontal sections the truck might have to pull forward before being able to turn. At this time, we only wanted reverse motion on the truck. Additionally, the horizontal section at the end of the path is to allow the truck to end with a zero angle on the trailer measured from the horizontal axis at the target.

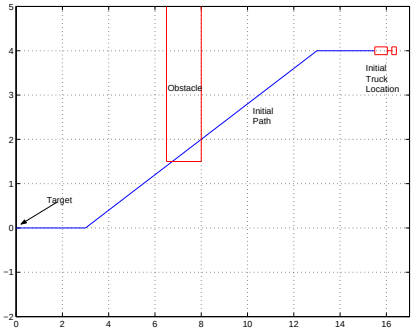

Figure 3: Initial conditions and initial path entered in the algorithm.

Given the obstacle locations, Level B, is able to generate a path around the obstacle. Figure 4 shows the path around the obstacle compared to the initial path.

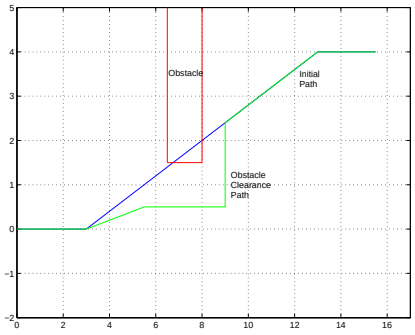

Figure 4: Obstacle avoidance path calculated by Level B.

The Obstacle avoidance path is then refined to eliminate the unnecessary turns. This simplified path is shown compared to the obstacle avoidance path in Figure 5. Level C eliminated the unnecessary turns created after the obstacle avoidance stage.

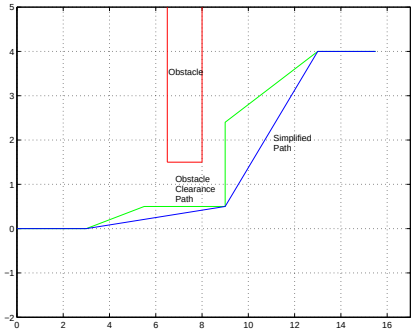

Figure 5: Simplified path calculated by Level C to eliminate unnecessary turns.

Figure 6 shows the smoothing of the simplified path. The sharp corners are rounded to create a path that the truck can follow. It is known that the truck can change approximately 1/2 of a degree in one time step. So the corners are smoothed to this rate of change on the angle of the path. The actual system will may not be able to turn at this rate depending on its speed.

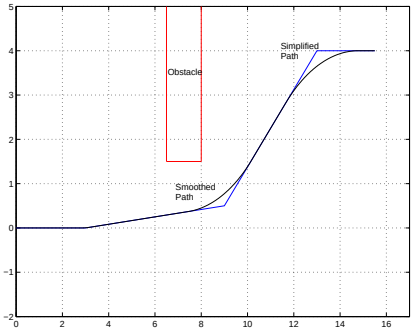

Figure 6: Smoothed path calculated by Level D from the simplified path.

The output of Level E is illustrated by Figure 7. The figure shows the smoothed path that the truck is to follow and the calculated angles of the truck at particular points. Only some of the points are shown in the figure to illustrate the motion of the truck.

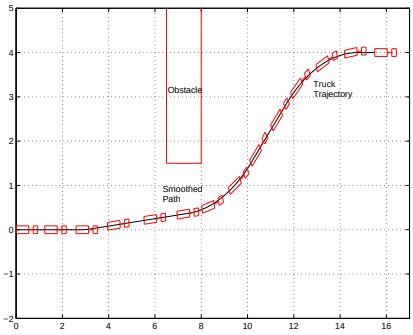

Figure 7: The truck trajectory path points calculated by Level E from the smoothed path.

Finally, the truck trajectory is applied to Level F and the actual system. An accurate model was created in order to do simulations of the system without the need of setting up the physical experiment [7]. Figure 8 shows the results when the trajectory was used as the input to the control system on-board the truck. The initial turn was actually too great for the truck to track so the truck slipped off the desired path. However, as the simulation continued, the truck returned to the path, and finished in the correct location at the target. Again, only some of the data is shown to illustrate the motion of the truck.

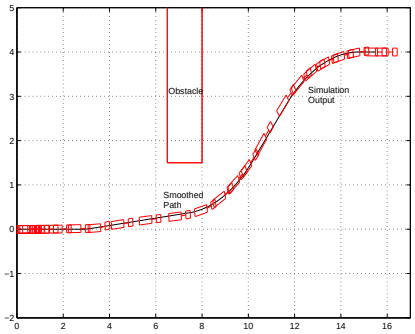

Figure 8: Simulation results of the truck as it follows the desired path.

A second experiment was run with the obstacle shifted further down. The same initial path was given to the system to see how it would react. Figure 9 shows the various outputs of the levels of the hierarchy from Level A to Level E. The main difference here is that the initial turn is a little longer, so the truck will miss-track again in this section, yet it is still within the region of convergence of the controllers. Additionally, the truck has to turn up again at the end to get back to the target. The truck also performs this maneuver successfully. The algorithm generated the new path in a just a couple of seconds.

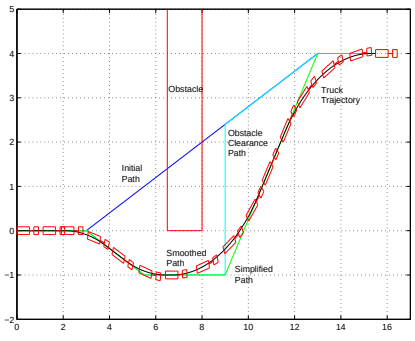

Figure 9: The outputs of the various levels of the hierarchy for a new obstacle location.

4 Conclusion

In this document we presented a hierarchical, obstacle-avoidance path-planner for use on a scale model of a tractor-trailer truck. The scale model was previously developed for other intelligent system experiments. In particular, for the autonomous control by a neural network path-planner.

The hierarchical approach helped to eliminate problems experienced in our previous work. The main advantage of the hierarchy is its modular construction. Small parts of the problem are able to be solved by the individual levels. The multi-resolutional aspect of the hierarchy is that each level has a complete picture of the situation to different degrees of detail and information.

The lowest levels have the most detail as to the actual motion of the truck, yet they lack the information about the goal or how to avoid obstacles. It is the job of the upper levels to provide the goal or any other difficult task. Since the problem is segmented, each task is much easier to solve. With just the simple algorithms we applied to the current system we are able to perform more difficult maneuvers, as well as in a much faster time, than we could with our previous neural network path planning.

We can now apply more robust intelligent agents in each level to try and improve the performance of the system. At present, the system is restricted to simple obstacles spaced a set distance apart. Given how this system is structured it will be easier to solve the problem of avoiding an obstacle. A system starting from scratch to solve the whole problem of path planning, obstacle avoidance, and control of the truck will undoubtedly have a much more difficult time and take far longer to reach a solution than our system.

References

[1] R. S. Woodley, T. Han, and L. Acar, "A new control scheme applied to the trailer-truck backing-up problem," in *Int. Engn. Sys. Through Art. N. Nets*, vol. 6, pp. 599–604, New York: ASME Press, 1996.

[2] R. S. Woodley, "The identification and a neural network based control for a backward maneuvering trailer truck," Master's thesis, University of Missouri-Rolla, August 1997.

[3] R. S. Woodley and L. Acar, "Non-linear system control using path generating neural networks," in *Int. Engn. Sys. Through Art. N. Nets*, vol. 8, pp. 599–604, New York: ASME Press, 1998.

[4] R. S. Woodley and L. Acar, "Neural network based control for a backward maneuvering trailer truck," in *Proc. 37th IEEE Conference on Decision and Control*, pp. 1611–1616, IEEE Press, 1998.

[5] T. Han and L. Acar, "A neural network based approach for the identification and optimal control of a cantilever plate," in *Proc. 1997 American Control Conference*, v. 1, pp. 232–236, June 1997.

[6] J. K. Antony and L. Acar, "Real-time nonlinear optimal control using neural networks," in *Proc. 1994 American Control Conference*, June-July 1994.

[7] R. S. Woodley and L. Acar, "A testbed system for nonlinear or intelligent control," in *Proc. 1999 American Control Conference*, pp. 3441–3445, IEEE Press, 1999.

[8] A. W. Divelbiss, *Nonholonomic Motion Planning in the Presence of Obstacles*. PhD thesis, Rensselaer Polytechnic Institute, 1993.

[9] D. Nguyen and B. Widrow, "The truck backer-upper: An example of self-learning in neural networks," in *Proc. 1989 IEEE International Joint Conference on Neural Networks*, pp. 357–363, 1989.

[10] B. Kosko, *Neural Networks and Fuzzy Systems: A Dynamical Approach to Machine Intelligence*. Prentice Hall, Inc, 1992.