Learning to Drive Among Obstacles

Bradley Hamner
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
Email: bhamner@andrew.cmu.edu

Sebastian Scherer
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
Email: basti@ri.cmu.edu

Sanjiv Singh
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
Email: ssingh@cmu.edu

***Abstract*— This paper reports on an outdoor mobile robot that learns to avoid collisions by observing a human driver operate a vehicle equipped with sensors that continuously produce a map of the local environment. We have implemented steering control that models human behavior in trying to avoid obstacles while trying to follow a desired path. Here we present the formulation for this control system and its independent parameters, and then show how these parameters can be automatically estimated by observation of a human driver. We present results from experiments with a vehicle (both real and simulated) that avoids obstacles while following a prescribed path at speeds up to 4 m/sec. We compare the proposed method with another method based on Principal Component Analysis, a commonly used learning technique. We find that the proposed method generalizes well and is capable of learning from a small number of examples.**

I. Introduction

We are interested in high speed operation of an outdoor mobile robot whose task is to to follow a path nominally clear of obstacles, but not guaranteed to be so. Such a case is necessary for outdoor patrolling applications where a mobile robot must travel over potentially great distances without relying on structure such as beacons and lane markings. In addition to avoiding obstacles, it is important that the vehicle stays near the designated route as much as possible. While the problem of detecting obstacles is itself challenging, here we consider issues related to collision avoidance while following a designated path given that the robot can detect obstacles in front of the vehicle in sufficient time to react to them.

Steering between obstacles is a difficult task, because good paths defy description by simple geometric constructs. Car-like vehicles, with a non-holonomic constraint, are limited in their capability to change direction, especially at high speeds where vehicle dynamics are a factor. Many methods of vehicle control have been reported in the literature, most of which rely on proper tuning of a set of parameters to a control function. These parameters dictate the robot's behavior, such as when to start avoiding obstacles, how much clearance to give obstacles, and relative level of desire to progress towards a goal while avoiding obstacles. The values of these parameters and their relation to each other are critical to the effectiveness of the algorithm. Insufficient repulsion from obstacles can lead to collisions. Too much obstacle repulsion can cause erratic behavior as the vehicle wildly veers away from obstacles. These gains have typically been tuned by hand over a large number of trials. The programmer decides when the vehicle's behavior is good enough.

We have implemented a model of collision avoidance based on studies with human subjects avoiding obstacles [1], [2]. The model proposed by Fajen and Warren is attractive because it uses a second-order control law, thus producing smooth paths, and is fast to compute. It also provides a means to predict where the vehicle will go if it follows the control scheme over some time horizon, implying a method to modulate the speed in a sophisticated way, as well as a means to detect cul de sacs that could trap the vehicle. While this control model holds the promise of high performance using a principled method, it still requires a large number of parameters to be adjusted to obtain a balance between safe and aggressive maneuvering.

We present a method for automatically learning the parameters of the control model. Using collected data from a human driving the vehicle, we employ machine learning techniques to tune the parameters to match simulated paths from the system with an operator's paths, while reducing oscillation.

II. Related Work

Collision avoidance for mobile robots is a fundamental technology, and a large number of methods have been developed for various applications. Most of the work has been indoors with vehicles that move at speeds where dynamics are not an important consideration. For example, Borenstein and Koren's vector field histogram (VFH) method transforms a local map into a one-dimensional discretized "polar obstacle density" function [3]. The angle closest to the heading to the goal that has low obstacle density is chosen.

Other notable, similar methods are the dynamic window approach [4] and the curvature-lane approach [5], which perform a parameter search in space of steering and velocity commands. These ideas have been extended in outdoor systems in which the robot projects candidate paths ahead of itself and then chooses among the corresponding steering actions the one that makes the most progress towards the goal and is obstacle free [6]. If no collision-free steering angle can move the robot towards a goal location, a higher level planner is consulted. Further work in the same vein uses a global planner in conjunction with the local planner and has been implemented on robots for space exploration [7], [8]. Both of these systems operate at low speeds (less than 1 m/s) where vehicle dynamics are not a factor.

Several systems have demonstrated off-road navigation. The Demo III XUV drives off-road and reaches speeds up to 10 meters per second. The speeds are high, but the testing

environments are rolling meadows with few obstacles. Obstacles are given a clearance which is wider than the clearance afforded by extreme routes. When clearance is not available, the algorithm plans at slower speeds [9]. Sandstorm and H1ghlander, robots developed for desert racing, have driven extreme routes at speeds up to 15 meters per second by planning in a series of grids along the original path and smoothing the result [10].

The model proposed by Fajen and Warren for collision avoidance seems similar to the standard potential field approach in that obstacles repel and the goal attracts but their are important differences. Rather than create a potential field over the state of the world, it stipulates a potential over the heading of the vehicle. Hence obstacles repel as a function of the difference between their bearing to the vehicle and the vehicle's heading. Second, the control is stated in a way that allows for the incorporation of vehicle dynamics.

Fajen and Warren's model was implemented on an indoor robot [1], [2], but the operation was simple – only a small number of point obstacles were considered and a single goal was specified. In contrast we would like our robot to drive outdoors at high speeds where it might encounter various configurations of obstacles, and since we would like the robot to track a specific path, the goal will move continuously. Recently Huang, et al., have used a modified version of the model proposed by Fajen and Warren that is geared towards obstacle avoidance using a monocular camera [11]. Since range to obstacles can not be measured directly, the width of obstacles (segmented in the image) is used instead of the distance. The authors report results with an indoor robot moving at 0.7 m/s.

There has been quite a lot of attention to learning as applied to mobile robots. Typically "learning" applies to perception as in learning a map of the environment [12] or learning a classification of the terrain given image or range data [13]–[15]. In a few cases learning is applied to the control itself as in a vehicle that learns to follow roads based on onboard video cameras while a human drives [16], [17]. We are not aware of other work where the robot learns to avoid obstacles based on observation of a human driver.

III. Approach

Here we present our model of collision avoidance based on the formulation by Fajen and Warren [2], and highlight the parameters to be learned by the system.

A. Control Model

Fajen and Warren's control model operates in the heading space of the vehicle. A defined goal point attracts the vehicle's heading, while point obstacles each repel the vehicle's heading.

In our situation, a desired path is given. We set the goal point to be a fixed distance, 10 meters, along the path from the vehicle's location. The goal's attraction in the control law increases as the angle to the goal increases. Since the goal is always nearly the same distance from the vehicle, we do not need a distance term in our control law (called MFW for modified Fajen/Warren):

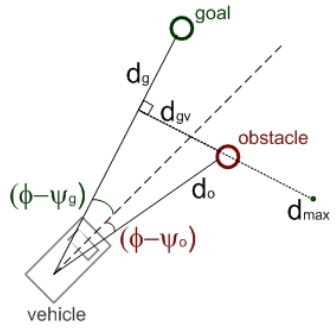

Fig. 1. Distance and angle terms used in the MFW control law. We consider the vehicle's position to be the center of the rear axle, so all distances are measured from that point.

$$\text{attract}_{\text{MFW}}(g) = k_g \cdot (\phi - \psi_g)$$

$(\phi - \psi_g)$ is the angle to the goal. d_g is the distance to the goal. k_g is a parameter which must be learned.

Since we use a single goal point at a large distance, vehicle path tracking is not highly accurate. However, our standards for path tracking accuracy are loose, and subordinate to the ability to avoid obstacles.

Similarly, each obstacle repels the vehicle's heading. The repulsion increases with decreasing angle and decreasing distance. Also, in curved path situations, an obstacle may lie directly in front of the vehicle, but far from its intended path. We therefore include a term which allows obstacles on the path to repel more. Then for each obstacle, there is a repulsion function:

$$\begin{aligned}\text{repulse}_{\text{MFW}}(o) = k_o(\phi - \psi_o)(e^{-c_3 d_o})\cdot \\ (e^{-c_4|\phi-\psi_o|})(1 + c_5(d_{\max} - \min(d_{\max}, d_{gv})^2)\end{aligned}$$

$(\phi - \psi_o)$ is the angle to the obstacle. d_o is the distance to the obstacle. k_o, c_3, c_4, and c_5 are parameters which must be tuned.

Calculating each obstacle's distance to the path is an expensive operation, as it requires finding the closest path point to each obstacle. Instead we make an approximation for the obstacle's distance from the path. We draw a vector from the vehicle to the goal point. An obstacle's repulsion is increased in proportion to its distance, d_{gv}, to that vector. If the distance is greater than $d_{\max}$, then no extra weight is applied. This maximum distance is based on the maximum distance the vehicle is allowed to be off the path.

Note that the MFW obstacle repulsion function actually gives zero repulsion for an obstacle directly in front of the vehicle. However, it is unlikely that an obstacle will ever be exactly at zero degrees to the vehicle. If it is at any small, non-zero angle, it will repel the vehicle away from zero degrees, and the repulsion will increase. The function is designed this way to make it easier for the vehicle to cross in front of an obstacle, so that if an obstacle is at a small angle to the vehicle, then other forces can push the vehicle to either side.

The goal attractions and obstacles repulsions are summed together, applying superposition. The result is a single control law for the vehicle's angular velocity:

Fig. 2. Our test platform is a modified all-terrain vehicle. A panning laser is mounted between the headlights. The fixed laser is mounted on top of the frame in the rear. The vehicle also carries two GPS antennae plus a differential antenna and an inertial measurement unit.

$$\dot{\phi}^*_{\mathrm{MFW}} = -k_g(\phi - \psi_g) + \textstyle\sum_{o \in O}(k_o(\phi - \psi_o)(e^{-c_3 d_o}) \cdot (e^{-c_4|\phi - \psi_o|})(1 + c_5(d_{\max} - \min(d_{\max}, d_{gv})^2)$$

We are left with five parameters in the control law, which can be expressed in a 5-tuple $\bar{u} = (k_g, k_o, c_3, c_4, c_5)$. See Fig. 1 for an illustration of the terms used in the MFW control law.

Note that superposition will not always yield good results. There are many situations in which attractions and repulsions can oppose each other in such a way as to cause the vehicle to not properly avoid obstacles. This is a limitation of the system. It is up to the user to find good parameter values to prevent these situations from occurring as much as possible.

B. Speed Control

Fajen and Warren found that most subjects walked at a constant pace, so they did not explore speed control. We constructed a speed control function based on the obstacle's distance and angle:

$$v = \min_{o \in O}\left[\frac{d_o}{2\cos(\phi - \psi_o)}\right]$$

This slows down the vehicle as obstacles get closer, which allows sharper turning and more time for the system to detect additional nearby obstacles. The 1/2 coefficient is included to ensure the vehicle will have over one second to stop for any obstacle. If the commanded speed is under .2 m/s, the system stops the vehicle.

For more detail of our control system, please see [18].

C. Vehicle Characteristics

All data collection and testing were performed on a modified all-terrain vehicle (ATV), as shown in Fig. 2. Obstacles were detected using data fused from two laser range finders, one fixed horizontally, and the other arranged vertically panning back and forth to cover the road. Vehicle pose information was provided by a pose estimation system, which uses the global position system (GPS) combined with motion data from an onboard inertial measurement unit (IMU) to produce positioning accurate to better than 5 cm.

To provide a better model of vehicle motion and increase the accuracy of vehicle simulations for the training and testing of parameters, we derived the vehicle's steering dynamics.

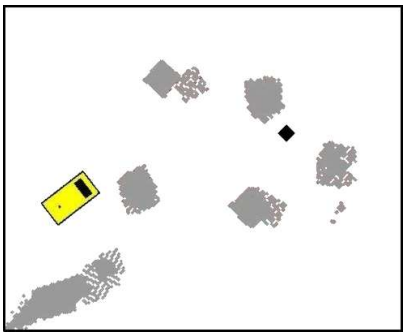

Fig. 3. The display to the driver during training. The small, black diamond on the right represents a moving goal point which slides along the desired path. The vehicle is drawn large to make its heading clear to the driver, but each obstacle is conservatively expanded in each direction by the length of the vehicle to ensure collision-free operation. A collision only occurs when the representative point of the vehicle intersects the "grown" obstacles.

The vehicle's steering actuator in contact with the ground behaves like a second-order system with delay. The identified parameters of the unit-mass spring-damper system are

$$\begin{aligned} \ddot{\theta} &= -b\dot{\theta} - k(\theta - \theta_d) \\ b &= 6.5789 \\ k &= 258.7 \\ t_{delay} &= 0.2s \end{aligned}$$

These values were used to generate vehicle paths as described in the following section.

IV. Search for Parameters

Tuning the set of parameters $\bar{u}$ by hand using intuition is tedious and difficult. In this section we outline a method to automatically determine the parameters based on a driver's behavior steering the robot.

A. Data Collection

A human subject drives the robot and tries to follow a path while avoiding obstacles. The path is replaced by a virtual goal point at a distance ahead of the current position as mentioned in Section III-A. Data about the goal point, the obstacles, and the driven path are recorded and used to determine the unknowns of the described control model. First we segment the path, and then optimize the parameters to match the subject's path.

The input to the control model and human subject at any point in time is a goal point p_g and a set of obstacles O defined as points in the x-y plane of the vehicle. A subject drives the robot only looking at a monitor which displays the virtual goal point and the relevant obstacles, not looking up to see the true location of obstacles ahead. Fig. 3 shows the only information available to the operator. The subject does know the goal point will remain at a near constant distance from the vehicle. Since goal distance is not a term in our control equations, we are unconcerned with the operator having this information. He has no other information regarding the path, or whether the goal will move to the left or right of the vehicle.

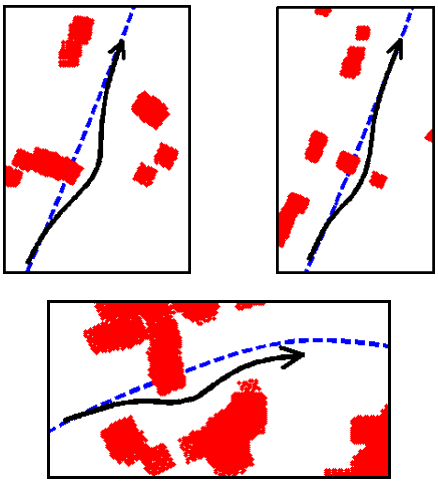

Fig. 4. A few examples of the data segments collected from the experiments with the operator. The dashed line portrays the desired path if there were no obstacles. The operator deviates from the path to avoid obstacles and returns when the path is clear. The parameters presented in Section V were learned using only these three segments as training data.

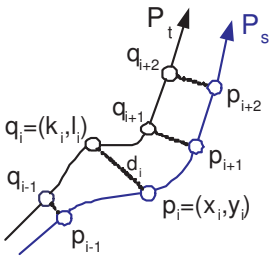

Fig. 5. The difference between two path segments is used to optimize the parameter set u. P_s is a recorded path segment while P_t was generated from simulating the system's driving using a parameter set u.

Using the goal and obstacle information, the operator drove the vehicle through a variety of paths and obstacle configurations. During operation, the pose estimation system collected information about the driven path at a rate of 100Hz. A pose data point contains a timestamp, an X-Y position, heading, velocity, and angular velocity. We also recorded the goal points given to the driver, as well as the locations of obstacles and when they were detected.

Often the paths were obstacle-free, where the driver simply followed the goal point. Of course, these situations provide no information with which to learn the obstacle parameters. We could learn the goal parameters alone, but we prefer to learn all parameters at one time. We therefore extracted only the relevant cases, breaking the data into short path segments consisting of cases where the driver steered the vehicle around the obstacles and returned to the path. A few of these segments are shown in Fig. 4.

B. Problem Setup

We use the path segments with obstacle avoidance behavior to train the parameters of our control model. Given a set $\bar{u}$ of parameters, we generate a path $P_t = \{q_i = (k_i, l_i) | i = 1..n\}$

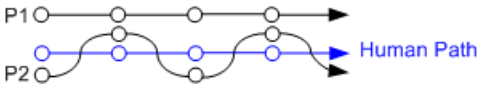

Fig. 6. Path 2 stays closer to the human path than Path 1, but is less desirable, due to its high frequency oscillations.

with the same n number of points as the training path segment P_s, which contains regularly sampled points in the plane. $P_s = \{p_i = (x_i, y_i) | i = 1..n\}$.

One measure of error is the Euclidean distance between each point pair as shown in Fig. 5: $d(\bar{u})_i = \sqrt{(k_i - x_i)^2 - (l_i - y_i)^2}$. However, we are also concerned with driving smoothly. Minimizing distance error alone could allow high frequency oscillations, illustrated in Fig. 6. Consequently, we add a term to penalize accelerations in the steering angle used by the autonomous system. The total error minimized between two path segments is $D(\bar{u}) = \sum_{i=1}^{n} [d(\bar{u})_i + |\ddot{\rho}_i|]$, where ρ is the commanded steering angle. We minimize the error of the parameter set for a number of path segments. Over m segments, the optimization procedure minimizes the combined error term $min_{\bar{u}} \delta(\bar{u}) = \sum_{j=1}^{m} D(\bar{u})_j$.

The path P_t is generated from a forward simulation of the steering behavior of the robot. This simulation is intended to be very accurate, using the steering model and its derived parameters presented in the previous section. Since the length of the path and velocities are not controlled in this model, we use the recorded speeds to ensure P_t has the same length as the training path P_s. The simulated path is generated as follows:

For a control iteration, the commanded angular velocity is

$$\dot{\phi}^*_{MFW} = -\text{attract}_{\text{MFW}}(g) + \sum_{o \in O} \text{repulse}_{\text{MFW}}(o).$$

The commanded steering angle given the translational velocity v and the vehicle length l is

$$\theta_d = \text{atan}(l \cdot \frac{\dot{\phi}^*_{MFW}}{v}).$$

The system retains a notion of the vehicle's current steering angle. The commanded steering angle is integrated with the current steering angle using the second order model presented in Section III-C.

$$\ddot{\theta} = -b\dot{\theta} - k(\theta - \theta_d)$$

The differential equations are integrated using the first-order Newton-Euler method to produce a new steering angle. The system then calculates the curvature of the arc the vehicle would travel on until the next control iteration.

$$\kappa = \frac{\tan(\theta)}{l}$$

Finally, the vehicle's position is updated to be at the end of that arc.

$$(x_{next}, y_{next}, \theta_{next}) = \text{updateArc}(x_{prev}, y_{prev}, \theta_{prev}, \kappa, v)$$

TABLE I

THE PARAMETER SETS

	k_g	k_o	c_3	c_4	c_5	err
Hand-Tuned	0.767	0.060	0.340	2.000	0.250	.3
Learned Random	0.8976	7.5537	0.9082	9.0856	0.5688	.18
Learned Genetic	0.6445	8.2175	1.6119	13.018	6.0817	.14
Learned SA	6.4328	8.7506	1.4971	5.0594	5.5838	.28

C. Learning Process

The relationship between the set of parameters $\bar{u}$ and the resulting paths is non-linear with many local minima. Normal gradient descent techniques are insufficient to find the global optimum, since they get trapped in the local minima. We address this problem using a two-step optimization process; first a nondeterministic algorithm to identify good candidates, followed by an optimization step.

We tried three different approaches for the nondeterministic step. The first is simply to randomly choose sets of parameters. We set the range of each parameter to be uniformly distributed between 0 and 10. With these ranges the system randomly picks 2500 sets of parameters. We keep the ten sets with the lowest residuals.

Another approach is a genetic learning algorithm. The system starts with a population of twenty-five randomly chosen sets of parameters, each parameter having a value between 0 and 10. Every iteration, twenty-five new sets are formed through combination and mutation. In combination, the system selects two parameter sets to form a new one, where each of the five new parameters is taken independently from one of the two parent sets. In mutation, the system selects one set and randomly mutates a number of its parameters to other values. The sets to be combined and mutated are drawn randomly with a distribution favoring those with the lowest residual, δ. The residuals of the population decrease over time as better sets are found. After one hundred iterations of the genetic algorithm (totaling 2500 error computations for a population of 25) we keep the ten sets with the lowest residuals.

The last approach is simulated annealing. The system starts with one randomly chosen set of parameters. At each iteration, the system either performs gradient descent on the current parameters or randomly selects new parameters. The probability of performing gradient descent increases over time, approaching one. We ran 2500 iterations of simulated annealing. From the 2500 sets whose residuals were calculated, we kept the ten with the lowest residuals.

Following the nondeterministic step, we applied a nonlinear least squares procedure with the previous best parameter sets as the initial guesses, producing ten optimized parameter sets for each approach. Then for each approach the optimized parameter set with the lowest residual, δ, was chosen as the best parameter set.

V. RESULTS

The methods described above learned parameter sets with only three training path segments, covering thirty meters of driving. We left the rest of our segments for test data.

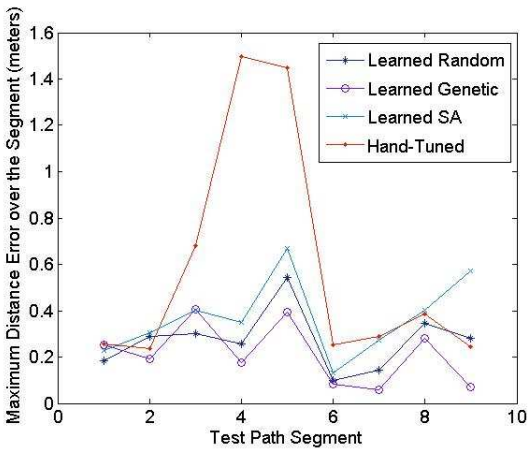

Fig. 7. The maximum distance error of the hand-tuned and learned parameters for each test segment.

Previously we used a parameter set hand-tuned over the course of a year. The set was tuned in a manner similar to gradient descent, starting with an initial guess and adjusting individual parameters slightly until we thought we had achieved the best vehicle behavior. The four sets are shown in Table I. Notice they are very dissimilar. We compare them on our remaining nine path segments, excluding the three training segments, in Fig. 7. This figure displays the maximum distance away from the human-driven path on each data segment. The hand-tuned parameters perform nearly as well as the learned ones in most segments, but are drastically far away in a few situations. A few of the test paths are shown in Fig. 8. The parameters learned with simulated annealing perform better than the hand-tuned parameters, but not as well as the other learned parameter sets. The parameters learned with the genetic algorithm perform only slightly better than the randomly learned parameters on the distance metric, averaging 6 centimeters closer to the human path, despite the difference in the learned parameters. The two have closer overall residuals due to more oscillation in the parameters from the genetic algorithm. As shown in Table I, the mean test residual is .3 for the hand-tuned set, .28 for the simulated annealing set, .18 for the randomly guessed set, and .14 for the genetically learned set. We chose the randomly learned parameter set to use for additional testing, since it drives the vehicle smoother than the genetically learned set, as shown in Fig. 9. Neither method gives high frequency oscillations, but the set learned by the genetic algorithm makes harder turns. This suggests that the terms we included in the error metric to ensure smooth steering need to be improved. Either smooth steering needs to be weighted higher, or we need to design a new metric to prevent those hard turns. All further results were obtained with the system using the set learned through random guesses.

Of course, our recorded data segments are only a small sample of what the vehicle may encounter. To ensure that the learned parameter set is better than the hand-tuned set, we also ran the system with the two sets in a vehicle simulator at 4 m/s, using new recorded obstacles from data files. Obstacle

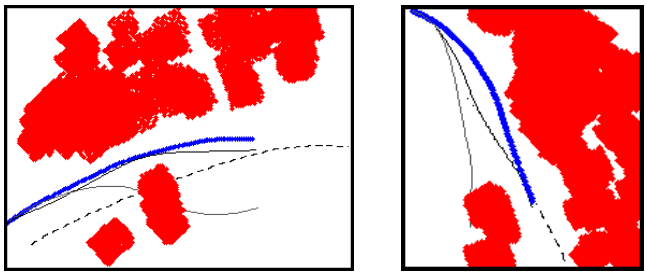

Fig. 8. The system's performance on test paths 4 and 5 with the two sets of parameters. The hand-tuned parameters (grey) cannot deal with the large obstacles on both sides of the vehicle. The randomly learned parameters (black) drive closer to the human's path (heavy line).

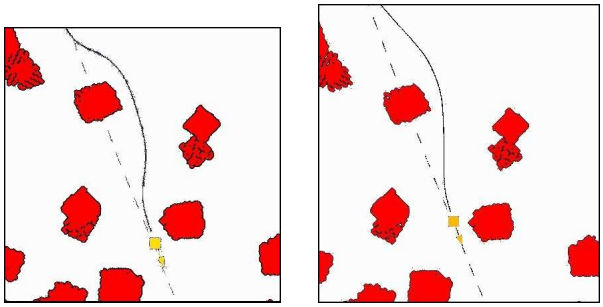

Fig. 9. At left, the path driven by the system using the parameters learned from the genetic algorithm. At right, the path driven by the system using the parameters learned by random guessing. They both avoid obstacles, but the genetic algorithm parameters make harder turns.

detection is simulated by an obstacle replay program which reads the vehicle's location from shared memory and serves all obstacles within a fixed distance from the vehicle to the simulator. Results from the simulator are shown in Fig. 10. The system using the hand-tuned set frequently gets the vehicle "stuck", where the vehicle has not avoided an obstacle and must instead stop before colliding with it. The system using the learned parameters gets stuck much less often. We have not encountered a case where the learned parameters get the vehicle stuck and the hand-tuned parameters do not. Also, recently, we have successfully used the learned parameters on the ATV test vehicle over a period of four months.

We have also implemented some classical road following algorithms to compare with our system. Specifically, we implemented eigenvector projection using principal component analysis developed for road-following using video imagery [17]. It is similar to steering using a neural network [16], but assumes a linear model of the control equation. We used all of our collected path segments as the training set, plus a copy of the set mirrored from left to right to remove the possibility of directional steering bias. The input images consisted of a map grid of the local area in front of the vehicle, with pixel values of -1, 0, and 1 indicating obstacle, free space, and road, respectively (shown in Fig. 11). To make the road, we expanded the desired path by two meters to either side. Each training data point supplied a vector containing the input image and a list of steering angle votes, which were Gaussian with the operator's steering angle as the mean. We stacked the vectors to form a matrix, and then calculated the eigenvectors

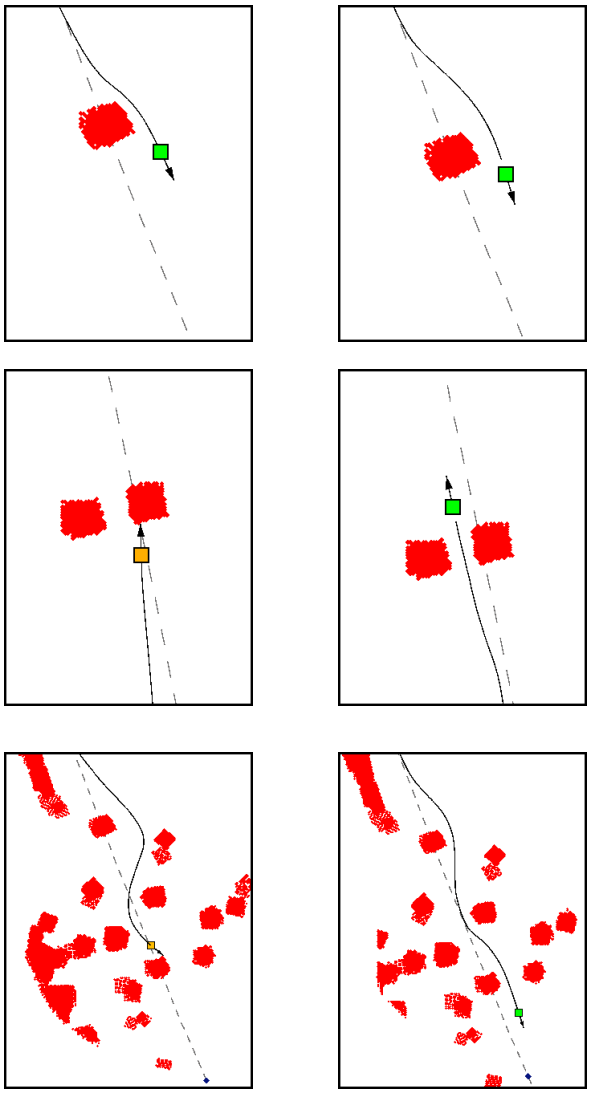

Fig. 10. We also tested the system in new situations at 4 m/s in simulation with recorded obstacles. At left, the hand-tuned parameter set can avoid individual obstacles, but performs poorly in more complex situations. The system stops the vehicle before a collision occurs, due to our speed control function. At right, the system avoids all obstacles when using the randomly learned parameters.

of that matrix.

Driving using the system consists of the following: an input image is taken. It is projected on to the image portion of the principal eigenvectors (in our case, the top ten eigenvectors). We then read the steering angle portion of that projection, which gives us a set of steering angle votes. We take the weighted average of the votes as the commanded steering angle.

We ran the PCA driving system on the path segments from its own training data. In general, the system steered the vehicle in the right direction, but consistently understeered. One resulting path is shown in Fig. 12. The simulated vehicle brushes the obstacles on both sides. We observe that to learn to drive using PCA the system needs much more training data than our own method requires. Especially, the vehicle does

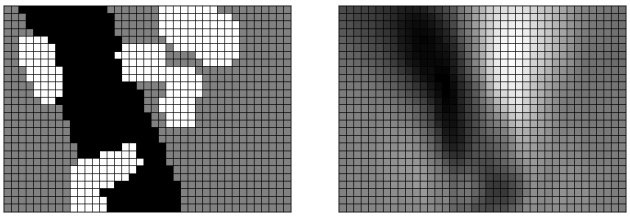

Fig. 11. At left, an image of a road with obstacles on it used as input to the PCA method. This input image comes from a local traversability map built using registered laser data. White regions represent obstacles, while dark regions denote the road. At right, the reconstruction of the image projected onto the ten principal eigenvectors.

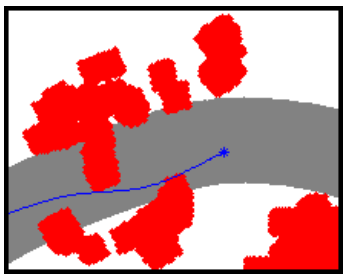

Fig. 12. Even in a situation seen in training, driving using PCA reduction does not avoid all obstacles. The grey stripe indicates the road.

not get close to obstacles in the training data, so the system using PCA does not know how to deal with such situations in practice.

VI. Conclusions

We modified an obstacle avoidance control law, originally modeled on pedestrian behavior, for a robot driving in a path-following situation through densely spaced obstacles. We have presented a method to automatically learn parameters for the control lased on path data from a driver. The system has proved able to control the vehicle smoothly while avoiding collisions.

Upon discovering that gradient descent minimizations performed poorly, we compared three randomized learning techniques. The genetic algorithm and random guessing performed similarly well. Simulated annealing performed worst, barely better than hand-tuning. We believe this is due to its being closely related to gradient descent. Once the system cools, the method essentially performs gradient descent on whichever parameter set it has, rarely jumping to a new set. One could have the simulated annealing method cool slower, but this would be akin to randomly guessing.

Since we only needed to tune the parameters of the control law, rather than learn a complete model of vehicle behavior, we were able to successfully teach the system how to drive using a small amount of data. Function approximators like principal component analysis need to see nearly every possible situation to learn how to drive the vehicle.

We did choose our training segments by hand from a larger set of recorded data. To make the learning truly automatic, the system needs to choose its own training data. It should not be difficult to determine when the vehicle is avoiding obstacles and extract the path segment from there. However, we have yet to implement it.

Also, we limited all parameters to a fixed range, even though the scales of each parameter are different, especially between the exponential decays and the other weights. Better results might be achieved by some analysis on the proper ranges of the parameters.

Our error metric was designed to both reduce the distance error and ensure the vehicle does not unnecessarily oscillate. However, after applying the learning method, we discovered we preferred less oscillation to a small improvement in distance. This suggests that our error metric should be adjusted to weight the two terms for this desired behavior, and furthermore that it could be adjusted to meet the preference of any user.

Many obstacle avoidance systems model vehicle control similar to the MFW system, where obstacle repulsion and goal attraction are modeled as potential functions. Our learning method relies only on simulating paths, not through analysis of how individual parameters affect the vehicle's steering. Therefore, we believe our method could be applied to teach other obstacle avoidance systems how to drive as well.

References

[1] B. R. Fajen,W. H.Warren, S. Termizer, and L. P. Kaebling, "A dynamical model of steering, obstacle avoidance, and route selection", International Journal of Comuter Vision, 54(1/2):13 V34, 2003.

[2] B. Fajen and W. Warren, "Behavioral Dynamics of Steering, Obstacle Avoidance, and Route Selection," Journal of Experimental Psychology: Human Perception and Performance, Vol. 29, No. 2, 2003.

[3] I. Ulrich and J. Borenstein, "VFH*: Local obstacle avoidance with look-ahead verification", In IEEE International Conference on Robotics and Automation, pages 2505 V2511, 2000.

[4] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance" IEEE Robotics and Automation Magazine, 4(1):23 V33, March, 1997.

[5] R. Simmons, "The curvature-velocity method for local obstacle avoidance", In IEEE International Conference on Robotics and Automation, volume 4, pages 2275 V2282, 1996.

[6] W. Feiten, R. Bauer, and G. Lawitzky, "Robust obstacle avoidance in unknown and cramped environments", In IEEE International Conference on Robotics and Automation, pages 2412 V2417, 1994.

[7] S. Singh et al., "Recent Progress in Local and Global Traversability for Planetary Rovers", Proceedings of the IEEE International Conference on Robotics and Automation, 2000, IEEE, April, 2000.

[8] C. Urmson and M.B. Dias, "Stereo Vision Based Navigation for Sun-Synchronous Exploration", Proceedings of the International Conference on Robotics and Automation, 2002, IEEE, May, 2002.

[9] P. Bellutta et. al., "Terrain Perception for Demo III", Proceedings of the IEEE Intelligent Vehicles Symposium 2000, Dearborn, MI, 2000.

[10] C. Urmson et al., "A Robust Approach to High-Speed Navigation for Unrehearsed Desert Terrain", to appear in the Journal of Field Robotics, 2006.

[11] Wesley H. Huang, Brett R. Fajen, Jonathan R. Fink, and William H. Warren, "Visual Navigation and Obstacle Avoidance Using a Steering Potential Function", Robotics and Autonomous Systems, 54 (2006) 288-299.

[12] S. Thrun, "Bayesian Landmark Learning for Mobile Robot Localization", Machine Learning, Volume 33 , Issue 1, October 1998.

[13] A Talukder, R Manduchi, R Castano, K Owens, L. Mathies, "Autonomous Terrain Characterisation and Modelling for Dynamic Control of Unmanned Vehicles", In Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems, 2002.

[14] C. Wellington and T. Stentz, "Learning Predictions of the Load-Bearing Surface for Autonomous Rough-Terrain Navigation in Vegetation", In Proceedings 4th International Conference on Field and Service Robotics, July1416, 2003.

[15] M. Hebert and N. Vandapel, "Terrain Classification Techniques from Ladar Data for Autonomous Navigation", Collaborative Technology Alliances conference, May 2003.

[16] Dean Pomerleau, "Neural Network Vision for Robot Driving", In The Handbook of Brain Theory and Neural Networks, Editor M. Arbib, 1995.

[17] J. Hancock and C. Thorpe, "ELVIS: Eigenvectors for Land Vehicle Image System", tech. report CMU-RI-TR-94-43, Robotics Institute, Carnegie Mellon University, 1994.

[18] S. Roth, B. Hamner, S. Singh and M. Hwangbo, "Results in Combined Route Traversal and Collision Avoidance", In Proceedings, International Conference on Field and Service Robotics, Pt. Douglas, Australia, July 2005.