Learning Maneuver Dictionaries for Ground Robot Planning

Pierre Sermanet[1,2] Marco Scoffier[1,2] Chris Crudele[2] Urs Muller[2] Yann LeCun[1]

(1) Courant Institute of Mathematical Sciences
New York University
New York, NY, USA

(2) Net-Scale Technologies
Morganville, NJ, USA

Abstract— **Vehicle dynamics is typically handled by models whose parameters are found through system identification or manually computed from the vehicle's characteristics. While these methods provide accurate theoretical dynamical models, they may not take into account differences between individual vehicles, lack adaptability to new environments and may not handle sophisticated models, requiring hand-crafted heuristics for backwards motion for example. Similarly to space and aerial maneuver-based planning methods, we demonstrate a simple and computationally fast planning method for ground robots with obstacle avoidance. It bypasses the need for model parameters identification and hand-crafted heuristics, learns the particularities of individual vehicles, allows on-line adaptation and sophisticated models. Human-driven or autonomously-driven trajectories are recorded and stored into a trajectory bank. While in learning mode, the robot records each traveled trajectory and places it into a bank, indexed by the initial speeds of each left and right wheels and the ending position at a fixed radius. Only the best trajectories are stored in the trajectory bank and then reused during autonomous runs for optimal short-range planning. Pre-computed (but not recorded) trajectories have been used in previous work and provide an important computational advantage over on-line computation methods, which are less practical in real-time applications due to the high-dimensional search space. A collision-free platform was developed without any hand-crafted heuristics or knowledge about the vehicle's characteristics. This method is demonstrated on the LAGR platform, a non-holonomic (differential drive) off-road mobile robot.**

I. Introduction

As part as the LAGR program (Learning Applied to Ground Robots), a software navigation platform was developed to support research in machine learning applied to long-range vision. In order to demonstrate its full potential, high-level and long-range perception must be free of the low-level perception, planning and control matters. Collisions and local planning optimization are mainly addressed by a low-latency system [16] that can take into account the vehicle's dynamics. To build this collision-free system, a multi-layered architecture was designed to split the long-range deliberative perception and planning [5], [17] from the fast and short-range reactive perception and planning. The reactive module perceives and plans in real-time with a low latency while taking into account dynamics. As part of a complete collision-free navigation platform, we propose a simple and computationally fast method to use sophisticated dynamics models while avoiding complex and innacurate modelling of traditional methods.

A. Related Work

Optimal control techniques [1] for motion planning provide optimal solutions but suffer from high computational costs, making them impractical for real-time applications. Most efforts have concentrated on curve-fitting for online planning from line segments [20] or arcs [13] or clothoids [9] or cubic spirals [8]. Curve-fitting relies on identification of the parameters of a vehicle's dynamical model. Our method provides a simple and automatic indentification of model parameters while bypassing the complex curve-fitting problem and resulting in a nearly null online computation cost. [15] introduced maneuver-based planning in free environments for aerial vehicles by recording human experts maneuvers, and later developed a maneuver-based automaton [3] that can concatenate trajectory primitives with a regular language. While the concept of trajectory recording for maneuver-based planning is the same, we extended it to perform obstacle avoidance as well, and apply it for ground robots non-holonomic planning. A similar pre-computed dynamics planning architecture was introduced by [2] but using an offline clothoids pre-computation method which still relies on hand-crafted modelling. [10] points out the need for a parametric trajectory representation in contrast to the prohibitively large space required by pre-storing methods. The practical results of [2] and this paper advocate in favor of trajectory pre-storing. While recording and pre-storing can certainly not address all nonholonomic planning problems, its simplicity and efficiency make it worth considering for some systems. Our recording method is currently limited to 2D planning and does not address the 3D challenges treated in [19], [7].

B. LAGR Research Context

Our long-range vision learning approach briefly introduced below is fully described in [5], [6]. The existing paradigm for vision-based mobile robots relies on hand-tuned heuristics: a stereo algorithm produces a (x,y,z) point cloud and traversability costs are assigned to points based on their proximity to a ground plane [11], [4]. However, stereo algorithms that run in realtime often produce costmaps that are short-range, sparse, and noisy. Our learning strategy uses these stereo labels for supervision to train a realtime classifier. The classifier then predicts the traversability of all visible areas, from close-range to the horizon. For accurate recognition of ground and obstacle categories, it is best to train on large, discriminative windows from the image, since larger windows give contextual information that is lacking in color and texture features. Other research has explored the use of online learning for mobile robots, but their methods have been largely restricted to simple color/texture correspondences [12], [14], [18].

II. Simple Trajectory Recording

The basic trajectory recording method described below provides good navigation performance with a minimum amount of development. Some refinements to increase performance will be described in the following section.

Fig. 1. The LAGR platform is a non-holonomic (differential drive) off-road mobile robot measuring 1.2m of length by 0.74m of width by 1.02m of height. It weighs 109kg and can travel at up to 1.3 meters/seconds. Despite its relatively small size, such a vehicle presents dynamics that cannot be ignored in a collision-free autonomous navigation system.

A. Motivation

The vehicle used in the LAGR program (Fig 1) presents dynamics that causes obstacle collisions if not incorporated in the reactive tactical planning. Dynamics is traditionally modeled using few model parameters found through parameter identification. Identification requires some knowledge about the vehicle's characteristics such as weight, dimensions, etc. In addition, the theoretical model of the vehicle does not account for specificities of individual vehicles. On the contrary, trajectory recording bypasses the need for parameters identification and can account for individual vehicle differences. Simple algorithms using theoretical models usually don't include complicated backward trajectories whereas any complexity of trajectory can be recorded without effort. Moreover, with its a great computational advantage over online curve-fitting methods, pre-storing is well suited for tight CPU budgets. To sum up, recording and pre-storing answer the needs of practical applications thanks to its simplicity of implementation, flexibility of use and computational efficiency.

B. Recording Trajectories

Trajectories can be recorded every time the robot moves, either in supervised mode (human driver) or self-supervised (autonomous). On the LAGR platform, information about the state of the vehicle is made available at 20Hz. This state includes among other information an absolute timestamp, the current pose and current speeds of each wheel. Similarly, wheel commands are sent at 20Hz to the motors in either manual or autonomous mode. These selected state variables along with the wheel commands are grouped into a *sample* every 50ms (20Hz). By recording this stream of samples, a feasible trajectory reaching the current position and the corresponding series of wheel commands are known at each timestamp. In other words, the radius R_{traj} to the current vehicle center (Fig 2) can be reached by looking back in the recorded sample stream and executing the wheel commands found in the stream. Every 50ms a new trajectory is produced and after some driving, enough trajectories are recorded to reach radius R_{traj} in all directions. The trajectory space has other input dimensions: the initial state $(LSpeed_0, RSpeed_0)$. Composed of both wheel speeds obtained from the vehicle state, the initial state allows the system to select the current set of feasible trajectories. When disregarding terrain differences (ice vs. asphalt), left and right initial wheel speeds are the main variables determining the current dynamics. For example in Fig 2, initial state $(4,9)$ means that the robot is making a hard left turn at $t = 0$. By analysing the trajectories going to the right, the figure makes clear that the dynamics of the vehicle have been captured. From this start state (turning hard left) in order for the vehicle to reach a candidate 90 degrees to the right it must make a large loop where left-turning and forward moving momentum is tranformed into a right turn. This curve has not been computed but simply selected, it is the fastest set of commands seen so far for getting the vehicle from a hard left turn to a hard right.

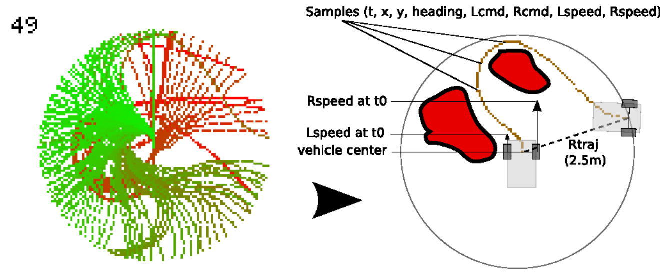

Fig. 2. **Trajectory recording and execution.** Trajectories are extracted from streams of samples recorded at 20Hz. A sample regroups information obtained from vehicle sensors (speeds and pose) with commands sent to motors. By looking ahead in the stream and recentering sample locations based on current location $(x, y, heading)_{t_0}$, feasible trajectories reaching the radius R_{traj} of 2.5m can be reconstructed. On the **left** are all best trajectories for state 49. Wheel speeds are approximately 0.2m/s for left ($LSpeed_{state} = 4$) and 1.3m/s for right ($RSpeed_{state} = 9$), thus the robot needs to execute large curves to reach candidates on the right. At planning time, the optimal trajectory is selected based on the current state and cost map (**right**), and executed using the series of (L_{cmd}, R_{cmd}).

Of course the accuracy of the trajectories depends on the accuracy of the pose sensor, which in the LAGR platform relies on a combination of the wheel odometers and an IMU. This pose information is assumed to stay accurate enough in the short term. If no visual odometry correction is available, it is preferable to record trajectories only on surfaces with low pose error rate (i.e. asphalt is preferable to ice). Once trajectories are extracted from the sample streams, they need to be sorted, compared and stored into the trajectory bank (pre-storing).

C. Trajectory Bank

The trajectory bank or maneuver library holds all best recorded trajectories. Each trajectory is indexed by $(LSpeed_0, RSpeed_0, Angle_{R_{traj}})$. $LSpeed_0$ and $RSpeed_0$ are the left and right speeds at time 0 of each trajectory. The LAGR vehicle drives at speeds ranging from -0.5m/s to 1.3m/s. This range is quantized into 10 different bins (0m/s lies in index 3). There are thus 100 different possible initial speed states. For each speed state, 160 $Angle_{R_{traj}}$ candidates are evenly divided around the perimeter at R_{traj} radius. In Fig 3, speed states along the diagonal have similar left and right wheel speeds at t_0, whereas at the top right and bottom left, the robot is steering harder right and left respectively. Because very hard turns are less frequent, these corners are more sparse than the area around the diagonal. A minimal bank with only 15% of all states filled (extracted from approximately 2 hours of human-driven recorded samples) showed great driving performance as demonstrated by experiment 9. The bank shown in Fig 3 is 64% full and was obtained using 18 hours of human-driven and autonomous-driven log-files (recorded during regular testing runs). The

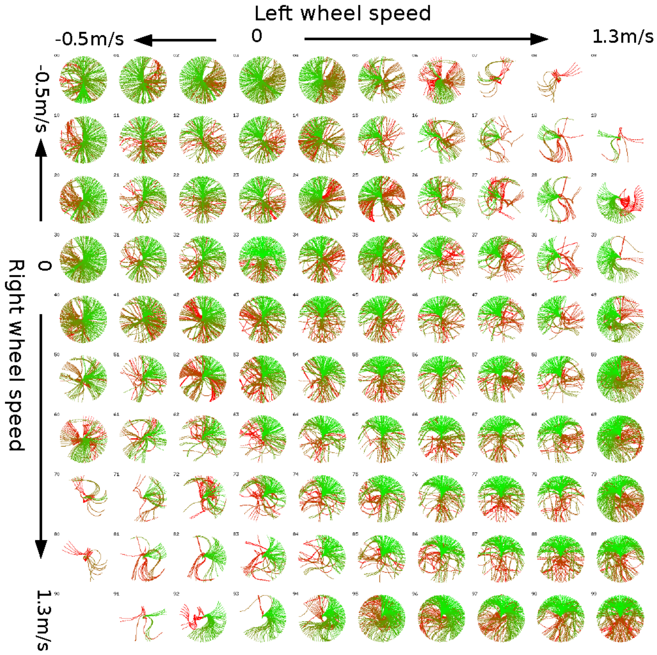

Fig. 3. **Trajectories bank (100 states)**. This graph shows all 100 states, each containing a set of 100 trajectories, each reaching out a different angle to a 2.5m radius around the vehicle. The horizontal and vertical axes represent the speed of the left and the right wheels. Each state is defined by a pair of initial velocities (v_{left}, v_{right}) at the vehicle center. States along the diagonal all have $v_{left} = v_{right}$, therefor in all these diagonal states the vehicle is initially moving straight forward. The absolute wheel speed difference $|v_{left} - v_{right}|$ increases when approaching the top right and bottom left corners, where feasible trajectories are grouped to one side. This bank is 64% full and was extracted from 18 hours of human and autonomous recorded runs.

latter bank shows similar performance as in the experiment 9 but can deal with a wider range of situations.

1) Bank Parameters Tuning: The discretization of wheel speeds and angular candidates as well as the radial distance of the trajectories does require some tuning based on the maximum driving speed of the vehicle and the available computational budget. The higher the maximum speed of the vehicle, the more wheel speed bins are required and the more the trajectory radius must be increased. With more resolution (more wheel states) in each input dimension, modeling accuracy increases but so do computational requirements. Moreover, the more states in the bank, the more trajectories need to be recorded. It was empirically found that on the LAGR platform, a minimum of 5 states per wheel was sufficient to obtain decent driving. 10 states provided the best results while keeping a rather small bank. The required number of angular candidates is estimated from the local map's resolution. With a cell size of 10cm and a radius of 2.5m, approximately 160 candidates $(2\pi * 25)$ are needed to have 1 trajectory per cell on the perimeter. The 2.5m radius was also chosen empirically and is based on the observation that most of the immediate dynamics effects fall within this range for this robot. It is better to keep the radius as small as possible to limit the trajectory sampling space for memory and bank-filling matters.

2) Scoring Criterias: Since in a race the vehicle is expected to minimize traveling time, the simplest scoring criteria for trajectory selection is the time it takes to reach the radius R_{traj}. When multiple trajectories are avaiable for a same candidate, only the one with the lowest score is kept. Different scoring formulas yielding additional trajectory types for increased navigation performance are described in further sections.

D. Planning with Trajectories

The first step in planning is to determine the current speed state from the motor sensors, i.e. the speed of each wheel. This speed state indexes from the bank the set of currently feasible trajectories (based on what we have recorded). Next, the set of trajectories is tested against the current costmap. All trajectories passing through non-traversable cells are ignored, the remaining set of possible trajectories are assigned a $cost_{traj}$ based on the cost of the traversed cells in the map and the recorded traveling time of the trajectory $time_{traj}$. Cell costs represent traversibility difficulty in seconds per meter. The minimum cost $cost_{min}$ of perfectly traversible terrain is $0.77 s/m$ (vehicle's maximum speed is $1.3 m/s$). Assuming trajectories are recorded on easy ground (e.g. asphalt), the minimum $cost_{traj}$ equals the sum of $time_{traj}$ and an "extra" cost given by the cells as follows:

$$cost_{traj} = time_{traj} + \sum_{i}^{n_{cells}} (cost_{cell_i} - cost_{min})$$

This formula augments the real recorded time of the trajectory by adding extra time based on the estimated difficulty of traversing a cell whose terrain has been classified by the perception modules.

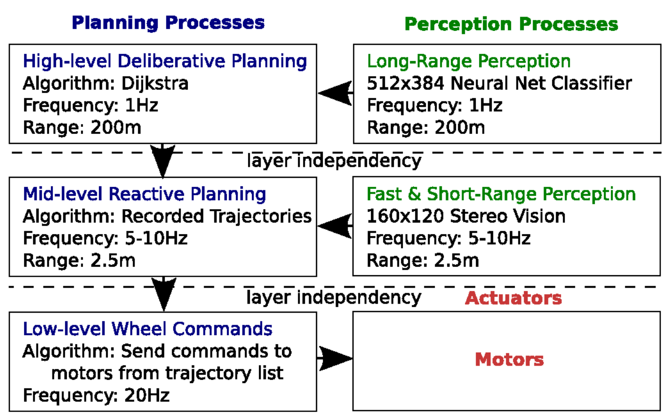

Fig. 4. **Multi-layer Perception and Planning Architecture.** Each level of planning and perception is protected against latencies and lower framerates in the higher levels, by being able to operate without getting updates from the higher more complex levels. For example, the low-level wheel commands process holds the list of motor commands to execute for the current trajectory and can operate independently until 2.5m away. It receives of course a new list much before reaching the end of it.

The trajectory costs are added to higher-level planning costs to form a global cost. The trajectory minimizing the global cost is selected and sent to a lower level planning execution process. This multi-layer perception and planning architecture (Fig 4 and [16]) protects each level from latencies in more complex (higher) levels. This allows us to lower the framerate and give more computation time to the higher and more complex levels. The top planning level is computed at 1Hz using a Dijsktra algorithm on a 200m radius hyperbolic-polar map described below and in [17]. It computes the optimal paths from the goal to each cell of the map and passes it on to the mid-level reactive planning which runs at 5 to 10Hz. Computing paths to all cells of the map is necessary because the mid-level perception and planning moves relative to the high-level map, and thus uses different interfacing cells at every mid-level iteration. Once the best trajectory is selected, a list of wheel commands is sent to the lowest-level process which is responsible for sending wheel commands at 20Hz. The resulting multi-layer mapping and planning is represented in Fig 5.

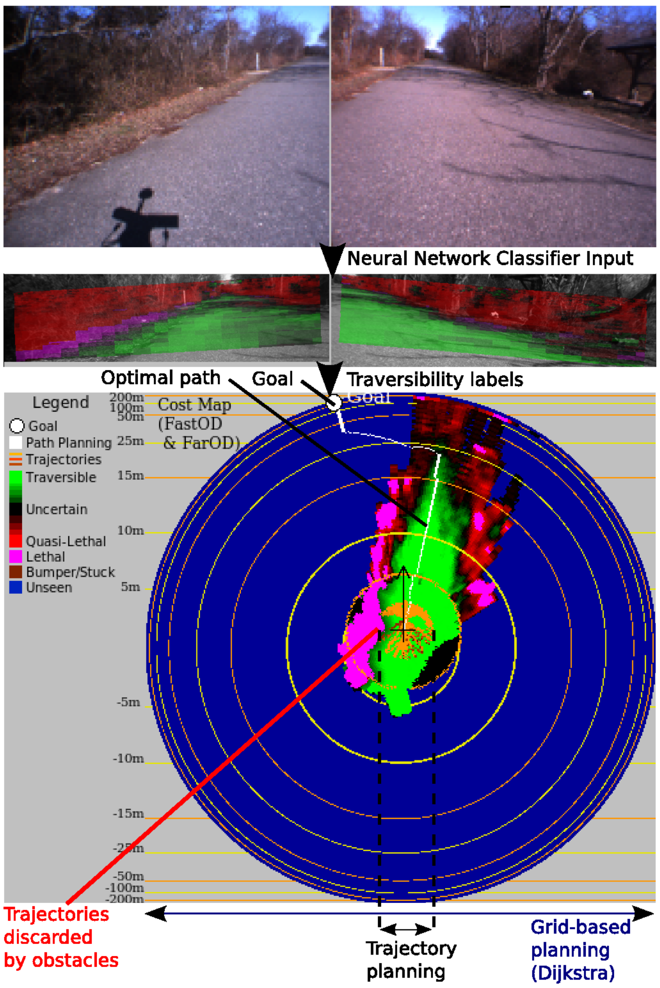

Fig. 5. **Multi-layer Planning with Trajectories.** Trajectories are first evaluted against the current (hyperbolic polar) map up to the 2.5m radius. Trajectories going through obstacles are discarded, others are given a cost based traversed cells costs and the trajectory's original traveling time. The cost of each feasible trajectory is then added to corresponding optimals path costs from the goal to all cells, as computed by the Dijkstra algorithm. Planning steps are executed at different paces, trajectories are run at 5-10Hz while Dijkstra is run at 1Hz. The globally optimal trajectory is finally sent to the low-level controllers and executed at 20Hz.

E. Tail whacking

To reduce computation, the width of the robot is taken into account by growing obstacles in the map by half the vehicle width. This simple method is fast but assumes that the length of the robot is null. Unaware of its length, the robot would often whack obstacles with its tail while turning. This issue is easily resolved by adding and keeping track of one or more points along the robot's length axis (Fig 6). A trajectory is discarded during planning if either the head or the tail trajectories encounter a non-traversable cell. With only one additional point near the tail of the LAGR robot, tail whacking completely disappeared during the numerous field tests.

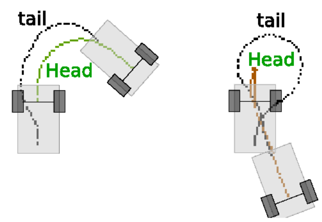

Fig. 6. **Tail whacking.** To account for the vehicle's length and to avoid hitting obstacles with the tail, one additional "tail" trajectory is computed from each recorded "head" trajectory. (Width is handled by obstacle growing.) During planning, a trajectory is ignored if either the head or the tail of a trajectory encounters a lethal cell. Multiple tail trajectories can be added depending on the vehicle's length, but only one tail trajectory was necessary to get rid of the tail whacking issue on the LAGR vehicle.

F. Fail Safe

The vehicle may fall into a state that has no recorded trajectory if it reaches a rare state (e.g. very hard turns), if the bank is not full enough or if the vehicle is completely boxed-in by obstacles. A simple solution is to stop the robot if it is moving or turning to let it fall into another state where there will be a useable trajectory. If it reaches the null speeds state and still no trajectory is available then, a backup mode is triggered (scripted driving straight backwards with a slight turn) to move the vehicle away from obstacles until feasible trajectories are available again. This simple solution has proved to be efficient as the vehicle is never stuck standing still nor stuck in backup mode in any of numerous field tests.

III. Refinements

The simple recording solution described above required minimal engineering and tuning efforts while providing a computationally cheap and collision-free navigation platform. Following are few additions to the system which brought further improvements. Other additions that increase the robot's self adaptability to changing environments are described in the further work section.

When planning with obstacles, having only one trajectory per initial wheel state and angular candidate can be restrictive. A single obstacle close the vehicle can wipe out all the trajectories in a general direction. To give more opportunites to the planner it is desirable to store multiple trajectories taking different paths to the same candidate. Storing multiple banks each with different criteria for selecting the trajectories one can achieve this goal. For example, one might record trajectories that take as straight a path as possible to the candidate, as well as trajectories which steer to the left or right before taking the fastest path to the candidate. This allows the planner to pick a trajectory which contours an obstacle on its way towards the optimal candidate for planning in the larger map.

By sampling only those trajectories which drive fastest from where the vehicle is to the 160 angular candidates around it, we are selecting the behavior we want from the vehicle (to drive fast) but also greatly reducing the space of all possible vehicle movements (all sequences of wheel commands) and other desireable behaviors. By adding other selection criteria we can more completely cover the space of possible movements, and produce more complex behaviors. For example in some situations, such as when the robot has made a wrong turn and gotten boxed in, it is desirable for the system to stop, turn in place and go straight rather than keeping momentum by driving as fast as possible along smooth curves. These different types of behaviors can be obtained simply by using different scoring measures when selecting trajectories from the recorded samples for a bank.

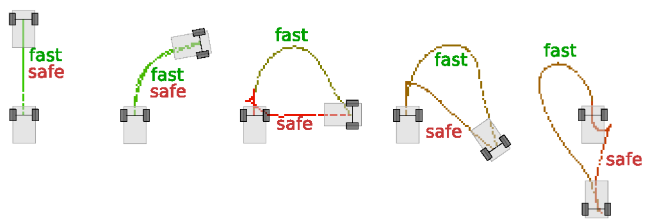

Fig. 7. **Fast and safe trajectories:** few examples of different trajectories obtained for a same end point using different scoring measures. The measure *fast* which rewards fastest trajectories, selected smooth trajectories with large curves, maximizing the vehicle's speed. The measure *safe* rewards maximum initial alignment of the vehicle's heading and the candidate heading and thus selected trajectories which cause the vehicle to stop and turn in place before driving straight to the candidate. Having dissimilar trajectories for a same candidate helps the vehicle to properly handle different situations.

Two types of trajectories were extensively tested and called *fast* and *safe* trajectories. The fast trajectories are selected by minimizing the time to reach the 2.5m radius and maximizing the distance traveled in the first few samples as follows. For all trajectories fulling a single slot in the bank $(LSpeed_0, RSpeed_0, Angle_{R_{traj}})$, retain the one that minimizes:

$$time_{traj} - \sum_{k=0}^{n_{cmds}-1} |k, k-1| * decay^k * \gamma_{dist}$$

where n_{cmds} is the number of wheel commands of the trajectory and $|k, k-1|$ the euclidian distance between two samples, measured by the vehicles wheel encoders. γ_{dist} is the normalization term between the time measure and the distance measure. By using the exponentiation of *decay* (Fig 8), we ensure that only the distance at the beginning is maximized. Because the fast planner picks a new trajectory every 100-250ms, in practice only the first few commands of a trajectory are actually driven therefore it is important to have a term which forces the desired behavior to happen at the beginning of the tragectory, hence the exponential decay terms.

Safe trajectories, on the other hand try to minimize the traveled distance and the angular difference between the vehicle's heading and the candidates heading, thus forcing the vehicle to drive as closely as possible to the ray from the vehicle to the candidate. To achieve this, the scoring formula minimizes the traveled distance while maximizing the initial turning toward the target heading $angle_{cand}$. The resulting behavior is turn on a dime and go straight in the candidate's direction. For all trajectories reaching a single $(LSpeed_0, RSpeed_0, Angle_{R_{traj}})$, retain the one that minimizes:

$$\sum_{k=0}^{n_{cmds}-1} |k, k-1| - |angle_{cand} - angle_k| * decay^k * \gamma_{heading}$$

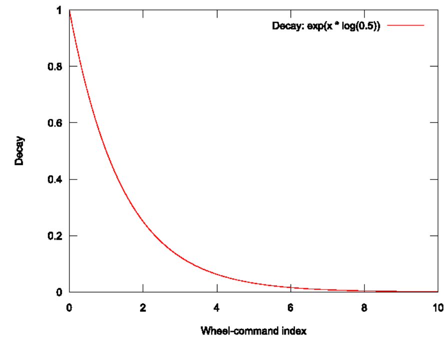

Fig. 8. **Trajectories types decays:** a decay of 0.5 is exponentiated with the wheel-command index in order to give more importance to first samples and near no importance after 5 samples for the distance and heading measure of the fast and safe trajectories.

where $angle_k$ is the heading of sample k. As in the fast trajectories, *decay* (Fig 8) helps maximize turning in the first few commands. $\gamma_{heading}$ is the normalization term between the distance measure and the heading measure.

IV. Results

We compare our system of learned vehicle dynamics to our previous system which uses a function of the steering angle to determine which wheel commands to apply. The vision, localization, mapping and planning of the two systems are identical, only the low level driving commands were changed. Our previous system performed adequately and was rated the first best performer in independent government tests at the end of Phase I of the LAGR project. But the lack of precise vehicle dynamics caused collisions which are completely avoided with the new system. Both systems were run several times over 5 different 30-meter courses, each consisting of a different arrangement of buckets as obstacles, with gradually increasing difficulty. Running times and number of bumper hits were recorded and averaged in Fig 9. As the difficulty increases, the running time and number of hits increases dramatically when using the old system (called "steering"). The most difficult course could not be tested because the system would systematically destroy the bucket arrangement. The new system with learned trajectories, on the contrary, keeps a low running time over all courses and a total of 0 collisions.

V. Limits

The following points were not actually limitating on the LAGR vehicle because of its relatively low maximum speed. However, some limits may appear with high speed vehicles or with greater number of input variables. Indeed, it is not clear what sampling precision is required for higher speeds and the trajectory bank size may become too large in memory and more difficult to fill as well. Increasing the number of input variables, such as road slippery estimation for example, would also greatly increase the memory requirements. Another limiting factor is the subsampling of the search space, which may be sub-optimal. It did not appear to be a limiting factor however in practice during testing. Moreover, similar pre-storing technique was used in [2] to successfully drive up to 35km/h, which suggest that pre-storing is still affordable at higher speeds. To sum up, while online trajectory search methods can give optimal solutions and do not require large memory, trajectory recording is bound by memory and space

Course	Running Time steering	Running Time trajectories	Obstacle hits steering	Obstacle hits trajectories	Number of Runs steering	Number of Runs trajectories
1 wall (easy)	19s	16s	0	0	1	1
2 walls (easy)	20s	15s	0	0	1	1
3 walls (moderate)	36.5s	18s	3	0	2	1
Slalom (hard)	56s	26.33s	11	0	1	3
Scattered (hardest)	X	24	X	0	0	1

Fig. 9. **Comparison of old steering-angle system (red) versus new learned-trajectories system (green).** Both systems were run 1 to 3 times over 5 different 30 meter courses each consisting of a different arrangement of buckets as obstacles with gradually increasing difficulty (**left:** slalom course). Running times and number of obstacle hits are reported. The hardest course could not be completed by the steering-angle system because the course was systematically destroyed by obstacle hits.
As the difficulty increases, the running time and number of hits of the steering-angle system increases dramatically. On the contrary, the new system with learned trajectories keeps a low running time over all courses and a total of 0 collisions. The trajectory bank used in this experiment was only 15% full and was extracted from 2 hours of human recorded runs.

subsampling. It gives however good approximations and very fast online computation and has not yet presented any of those limitations in practice.

VI. Further Work

Besides exploring more difference measures to cover more of the trajectory search space as discussed above, another interesting way of research is to increase the autonomy of trajectory recording with online and offline methods, in order to facilitate even more the recording process without any human intervention and to constantly optimize trajectories during navigation.

1) Offline Trajectories Search with Global Search Heuristics: As in [3], a set of recorded maneuver primitives could be used as a basis for a more complicated “language” of trajectories. An offline algorithm could reconstruct new non-recorded trajectories from pieces of those basic trajectories. This way, all slots of the bank could get filled and optimized more easily. To speed up the search of optimized trajectories, global search heuristics such as genetics algorithms could be used to take advantage of the best trajectory pieces combinations.

2) Online Trajectories Recording in Production Mode: When the vehicle runs in production mode, it is very cheap to record and add new trajectories on the fly when more optimized ones occur. With minimal effort, the robot can constantly improve its driving skills through experience.

3) Online Trajectories Recording and Search in Self-supervised Mode: Given a large obstacle-free training area, the robot can try to fill its trajectory bank itself by knowing which areas need trajectories or optimization. During this self-supervised learning mode, the trajectory space could be searched using simple wheel commands heuristics and gradient descent to approach the desired target points.

VII. Conclusion

We have described a simple and efficient trajectory recording method as part of a reliable collision-free navigation platform. The LAGR vehicle dynamics was integrated without any complex model parameter identification nor online curve-fitting solution search and with minimal engineering effort.

Acknowledgements

This work was supported in part by the DARPA LAGR program under contract HR001105C0038.

References

[1] A. E. Bryson and Y. C. Ho. Applied optimal control. In *New York: Hemisphere Publishing*, 1975.
[2] D. Coombs, K. Murphy, A. Lacaze, and S. Legowik. Driving autonomously offroad up to 35 km/h. In *Intelligent Vehicles Symposium*, 2000.
[3] E. Frazzoli, M. A. Dahleh, and E. Feron. Maneuver-based motion planning for nonlinear systems with symmetries. In *IEEE transactions on robotics*, 2005.
[4] S. B. Goldberg, M. Maimone, and L. Matthies. Stereo vision and robot navigation software for planetary exploration. In *IEEE Aerospace Conf. Proc.*, March 2002.
[5] R. Hadsell, A. Erkan, P. Sermanet, M. Scoffier, U. Muller, and Y. LeCun. Deep belief net learning in a long-range vision system for autonomous off-road driving. In *Proc. of Int'l Conf on Intelligent Robots and Systems (IROS)*, 2008.
[6] R. Hadsell, P. Sermanet, J. Ben, A. Erkan, J. Han, B. Flepp, U. Muller, and Y. LeCun. Online learning for offroad robots: Using spatial label propagation to learn long-range traversability. In *Proc. of Robotics: Science and Systems (RSS)*, 2007.
[7] T. Howard and A. Kelly. Terrain-adaptive generation of optimal continuous trajectories for mobile robots. In *International Symposium on Artificial Intelligence*, 2005.
[8] Y. Kanayama and B. Hartman. Smooth local path planning for autonomous vehicles. In *Technical Report, Dept. of Computer Science, University of California, Santa Barbara*, 1988.
[9] Y. Kanayama and N. Miyake. Trajectory generation for mobile robots. In *Robotics Research, MIT Press, Cambridge*, 1985.
[10] A. Kelly and B. Nagy. Reactive nonholonomic trajectory generation via parametric optimal control. In *International Journal of Robotics Research*, 2003.
[11] A. Kelly and A. Stentz. Stereo vision enhancements for low-cost outdoor autonomous vehicles. *ICRA Workshop WS-7*, May 1998.
[12] D. Kim, J. Sun, S. M. Oh, J. M. Rehg, and B. A. F. Traversibility classification using unsupervised on-lne visual learning for outdoor robot navigation. In *IEEE Int'l. Conf. on Robotics and Automation (ICRA)*, May 2006.
[13] K. Komoriya, S. Tachi, and K. Tanie. A method for autonomous locomotion of mobile robots. In *Journal of the Robotics Society of Japan, vol 2, pp 222-231*, 1984.
[14] R. Manduchi, A. Castano, A. Talukder, and L. Matthies. Obstacle detection and terrain classification for autonomous off-road navigation. *Autonomous Robot*, 18:81–102, 2003.
[15] M. PiedMonte and E. Feron. Aggressive maneuvering of autonomous aerial vehicles: A human-centered approach. In *International Symposium on Robotics Research*, 1999.
[16] P. Sermanet, R. Hadsell, J. Ben, A. N. Erkan, B. Flepp, U. Muller, and Y. LeCun. Speed-range dilemmas for vision-based navigation in unstructured terrain. In *Proc. 6th IFAC Symposium on Intelligent Autonomous Vehicles*, 2007.
[17] P. Sermanet, R. Hadsell, M. Scoffier, U. Muller, and Y. LeCun. Mapping and planning under uncertainty in mobile robots with long-range perception. In *Proc. of Int'l Conf on Intelligent Robots and Systems (IROS)*, 2008.
[18] B. Sofman, E. Lin, J. Bagnell, N. Vandapel, and A. Stentz. Improving robot navigation through self-supervised online learning. In *Proc. of Robotics: Science and Systems (RSS)*, June 2006.
[19] M. Spenko, Y. Kuroda, S. Dubowsky, and K. Iagnemma. Hazard avoidance for high-speed mobile robots in rough terrain. In *Journal of Field Robotics*, 2006.
[20] T. Tsumura, N. Fujiwara, T. Shirakawa, and M. Hashimoto. An experimental system for automatic guidance of a robotic vehicle following a route stored in memory. In *Proceedings of the 11th International Symposium on Industrial Robots, pp 187-193*, 1981.