2009 IEEE International Conference on Robotics and Automation
Kobe International Conference Center
Kobe, Japan, May 12-17, 2009

Smooth Path Planning in Constrained Environments

Martin Rufli[†] Dave Ferguson[‡] Roland Siegwart[†]

[†]*Autonomous Systems Lab, ETH Zürich*
Tannenstrasse 3
8092 Zürich, Switzerland
martin.rufli@mavt.ethz.ch r.siegwart@ieee.org

[‡]*Intel Research Pittsburgh*
4720 Forbes Ave
Pittsburgh, PA, 15213, USA
dave.ferguson@intel.com

***Abstract*— In this paper we describe a novel path planning approach for mobile robots operating in indoor environments. In such scenarios, robots must be able to maneuver in crowded spaces, partially filled with static and dynamic obstacles (such as people). Our approach produces smooth, complex maneuvers over large distances through the use of an anytime graph search algorithm applied to a novel multi-resolution state lattice, where the resolution is adapted based on both environmental characteristics and task characteristics.**

In addition, we present a novel approach for generating fast globally optimal trajectories in constrained spaces (i.e. rooms connected via doors and hallways). This approach exploits offline precomputation to provide extremely efficient online performance and is applicable to a wide range of both indoor and outdoor navigation scenarios. By combining an anytime, multi-resolution lattice-based search algorithm with our precomputation technique, globally optimal trajectories in up to four dimensions (2D position, heading and velocity) are obtained in real-time.

Fig. 1. Herb, the human assistance robot at Intel Research, Pittsburgh. A video of Herb navigating smoothly through the lab accompanies this paper.

I. INTRODUCTION

Autonomous vehicles are often required to navigate through cluttered, dynamic, and unstructured environments. Due to various vehicle and environmental constraints, they often need to plan complex maneuvers to perform this navigation. Further, to provide responsive vehicle behavior in dynamic environments, the time spent performing this planning is required to be minimal.

Historically, fast and safe planning has been achieved via *local planners*, which perform short-term reasoning for a robot considering only the local vicinity of the environment. Such approaches include potential field-based techniques [1] on one hand, where a gradient guides the robot to a designated goal pose while avoiding high potential obstacles, and curvature velocity [2] and dynamic window [3] based approaches, where dynamically feasible local actions are computed in control space. One of the major limitations of these approaches is their susceptibility to local minima, such as dead ends, due to the limited range of their planning. To overcome these issues, planners incorporating global as well as local information have subsequently been developed.

Global planners construct a plan for the robot from its current position all the way to its desired goal. In this case, deterministic graph search algorithms such as Dijkstra [4] or A* [5] search have been successfully applied to low dimensional representations of the navigation problem, where (uniform) grid-based sampling is often used due to its simplicity. Paths returned by such global grid based algorithms, although optimal on the grid, are often not achievable due to vehicle constraints. Other issues lie in the limited number of headings (i.e. increments of 45 degrees) standard grid-based planners allow and their prohibitive execution times in higher dimensional search spaces. Various efforts have been made to overcome these shortcomings, including increasing the available headings or approximating continuous paths (see [6] and [7]), using suboptimal variants of the search algorithms to reduce computation time [8], and efficiently updating existing solutions when new information is received [9], [10].

However, despite the large body of existing work on long-range navigation, real-time, deterministic, global planning in high dimensional spaces (required for smooth navigation) remains very challenging. It is thus not surprising that navigation systems are usually still equipped with a local planner complementing the global one, providing smooth local behavior while biasing movement based on the global plan.

The main contribution of this paper is a navigation approach that produces smooth, complex maneuvers over large distances. This is achieved via the use of an anytime graph search algorithm (Anytime A* [8]) applied to a novel multi-

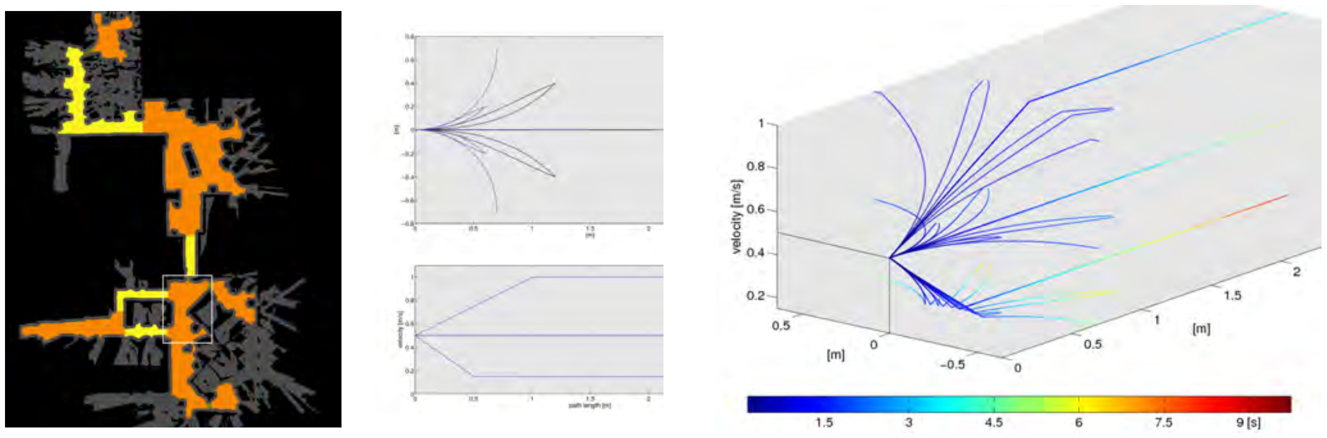

Fig. 2. Left: the Intel Research Pittsburgh lab as an example of a segmented environment (segmented manually), producing rooms (orange), hallways (yellow), and doors (olive). Segmentation is achieved through either hallway entries, or doors. Center and Right: example of a 16-directional multi-resolution lattice (one of the 16 initial headings shown). Top Center: black segments denote the low-resolution part. Black and blue segments together denote the high-resolution part. Some of the short straight segments are occluded. Bottom Center: velocity profiles as a function of path length. Note, that deceleration is higher than acceleration capability. Stops from 0.15[m/s] to standstill are commanded instantly. Right: full 4D (x, y, θ, v) view for one heading.

resolution state lattice, where the resolution is determined based on both environmental characteristics (e.g. narrowness) and task characteristics (e.g. proximity to goal). In addition, we present a novel approach for generating fast globally optimal trajectories in constrained spaces (i.e. rooms connected via doors and hallways). This approach exploits offline precomputation to provide extremely efficient online performance and is applicable to a wide range of both indoor and outdoor navigation scenarios. The combination of these contributions results in a novel navigation approach that exhibits global optimal planning capability in 4D (x, y, θ, v), guaranteeing smooth trajectory following when employed in constrained areas, and/or large, sparse, and open spaces. When these conditions are not met, planning over 4D (x, y, θ, v) locally ensures smooth trajectory following. An anytime property allows for good intermediate solutions and continual solution improvement as deliberation time allows.

In Section II, we introduce our novel multi-resolution state lattice. Section III explores two approaches for reducing the complexity of the state space and Section IV provides simulated and physical results from our planner operating in constrained indoor environments.

II. SEARCH OVER A STATE LATTICE

Grid-based search (usually 2D) is a common method for reducing the complexity of an environment, making it computationally tractable for real-time planning. Such searches do not, however, respect vehicle kinematic nor dynamic constraints due to the limited representation of the grid. Post-processing steps are thus often required. Planning in higher dimensions (e.g. incorporating velocity and time) has recently been advocated to circumvent such postprocessing steps, yet in such cases an even more drastic reduction in search space is required to retain real-time capability.

A representation that addresses this limitation comes in the form of the state lattice, a construct that reformulates the nonholonomic motion planning problem into a graph search and may reduce the search space in an intelligent way so as to retain compliance with vehicle kinematic and dynamic constraints [14]. Specifically, a suitable vehicle model could be forward simulated in time using desired translational and rotational velocity as model inputs. These input sequences could then be stored together with the resulting trajectory primitive (lattice segment) they represent. Compliance with vehicle kinematic and dynamic constraints can thus be guaranteed, given the above design iteration and a high-fidelity dynamic model of the vehicle.

In our implementation we took a simpler approach, which does not fully respect vehicle kinematic constraints: we used cubic splines, represented via the Catmull-Rom parametrization, re-sampled for constant path length. This allows us to control each lattice segment's initial and final position and heading, as well as limit the maximal curvature along the trajectory. Maximal curvature and acceleration were chosen experimentally in agreement with the capabilities of the platform (Segway RMP, see Fig. 1).

The state lattice is then constructed by starting with a node in a desired configuration space (in our case 4D: (x, y, θ, v)) and then creating edges emanating from this node (see Fig. 2). From every node these edges transition to, the process is repeated (and so on until the desired bounds of the lattice have been met), resulting in a connected graph consisting of all the nodes and edges generated. Typically, a discretization is applied to the nodes so that they all reside on some grid in the (potentially high-dimensional) configuration space. Refer to [14] for an automated such approach.

A. Multi-Resolution Lattices

A key factor when designing a state lattice is the resolution of the discretization used to represent the nodes in the lattice in terms of the *position discretization* (which we denote the grid) the *heading discretization*, and the *velocity discretization*.

From a *computational* point of view, the underlying position discretization should be chosen as coarse as possible. From a *completeness* point of view, it needs to be fine enough to generate feasible plans through hallways and doorways. Specifically, the robot diameter should be an *uneven* multiple of the grid resolution. In this way, the robot center can be made to lie on a grid point. For our application (indoor navigation with a Segway RMP), a 0.1[m] grid was found to satisfy these constraints.

State lattices have been successfully employed with 8 to 64 (uniform) directions in both outdoor and indoor environments. We found that for smooth *and* natural looking paths in indoor environments, generally at least 16 directions are needed. In special cases (such as described in Sec. III), as low as 8 directions might be sufficient. If frequent interactions with dynamic obstacles are anticipated and/or movement in narrow 'oblique' hallways is required, a finer angular discretization is advantageous. 16 directions were found to work well for our indoor office environment. Expanding 16-directional lattice segments over a 0.1[m] grid is computationally intensive, however, and in many cases (such as in wide open spaces) not necessary to achieve a smooth and feasible trajectory. A recently proposed solution to this problem is to incorporate a *multi-resolution* lattice, operating at two resolutions based on task characteristics [11]. The lower resolution is usually chosen as a subset of the higher resolution lattice so that suboptimality guarantees can be given with respect to the low-resolution lattice. We extend this multi-resolution approach by considering both environmental *and* task characteristics in a novel way for the position.

We also incorporate velocity into the lattice state space, to produce trajectories instead of paths during the planning stage, rendering postprocessing steps superfluous. Minimal cost solutions are thus time and no longer path length optimal. In order to obtain real-time performance, we currently employ a coarse velocity discretization: fast (1.0[m/s]), medium (0.5[m/s]), slow (0.15[m/s]) and standstill (0.0[m/s]). 0.15[m/s] is the lowest steady velocity we found the Segway RMP to be capable of executing. Standstill can thus be commanded instantly, as soon as 0.15[m/s] is reached. In the future, we plan on increasing the number of allowed velocities in a multi-resolution approach, based on task characteristics (i.e. close to the current robot position).

The currently implemented 4D multi-resolution lattice is depicted in Fig. 2.

1) Environmental characteristics: In narrow areas, high-resolution lattice segments are often required to find feasible, non-meandering paths and the velocity of the robot along these segments should be kept low (i.e. 0.5[m/s]) due to decreased flexibility for safety maneuvers. We therefore propose to expand high resolution lattice segments and enforce lower velocities in narrow areas of an environment. Narrow areas can be detected by the application of a bridge test [13].

2) Task characteristics: Both the goal and start locations of a search have special significance. It is often of interest to approach a goal accurately and a higher resolution lattice helps in such cases. By also considering high resolution segments around the start position of the search (i.e. the robot's current position) in conjunction with frequent re-planning, the robot is guaranteed to always remain on a high fidelity trajectory. These considerations have the effect of generally producing trajectories close to the optimal one (had only the high-resolution lattice been expanded) and at the same time generating solutions much faster than by expanding the high-resolution lattice alone [11].

B. Search over the Lattice

A state lattice can be considered a specific method for constructing a directed graph. Thus, regular graph search algorithms are applicable for searches along a state lattice. The most popular and efficient deterministic graph search algorithms belong to the family of 'best-first' searches. They expand promising states first, making use of a heuristic function to guide them, the original one being A* [5]. Subsequently, more efficient algorithms (particularly for re-planning) have been devised (see i.e. [12] for an overview). If there are several dynamic obstacles (such as people) in the environment, however, their constantly changing trajectories render incremental re-planning algorithms less effective, so that re-planning from scratch with A* is preferable. We use the Anytime A* extension [8] in 4D (x, y, θ, v) over the above presented multi-resolution state lattice.

1) Heuristic generation: An informative heuristic is an important component of the high dimensional lattice search. We use a 2D search that is performed (on the static map) once for every new goal selected for the robot (as it is dependent on the goal location alone, and thus does not change). Since goals typically do not change for several seconds, such a calculation is allowed to take longer than the usual sampling time requirements for the planner (especially if the robot is stationary). We perform a Dijkstra search out from the goal location(s) so that the cost values become the heuristic values for the high-dimensional search. To ensure this heuristic is admissible, we constrain the costs of actions to be upperbound by their Euclidean distance. In particular, the obtained heuristic costs are divided by 1.03 for the 16-neighborhood employed (see [15] for details).

2) Representation of dynamic obstacles: Representing dynamic obstacles for planning remains a challenging problem. In our indoor environment, the only dynamic obstacles are humans operating at comparable velocities as our robotic platform. We currently detect them via laser readings and store them in the same way as static obstacles. Operation in open areas in combination with frequent re-planning (i.e. at 10Hz) guarantees successful navigation. Our approach faces limitations in narrow hallways, however, where dynamic obstacles may block the only path to the goal. No solution is thus found in such cases and the robot must wait until the person has moved before continuing. The introduction of time into the search space, in conjunction with a short-term prediction of the dynamic obstacles' future movement could solve this problem in the future.

III. REDUCTION OF SEARCH COMPLEXITY

By using a multi-resolution state lattice, vehicle kinematic and dynamic constraints can be considered during the search while at the same time a substantial reduction in search complexity can be obtained. By planning in 3D (x, y, θ) over a multi-resolution lattice, performance similar to 2D (x, y) grid planning has been reported [14]. Recently, however, it has been of great interest to increase search dimensionality even further to produce smoother, faster motions or avoid dynamic obstacles. In such cases, obtaining real-time re-planning performance is extremely challenging due to the increased dimensionality of the search.

However, often the environment itself is constrained in ways that can be exploited to reduce the computational complexity of the search problem. Two examples of such constraints are described below.

A. Orthogonal Structures

Outdoor environments do not usually exhibit regular geometric features. In indoor environments, on the other hand, walls are usually oriented orthogonally to each other. In such areas, the lattice angular discretization may be reduced: four discrete values would theoretically suffice, with the lattice being made up of only straight line segments and quarter-circular arcs. When interacting with obstacles, however, 8 or 16 directions might be a more sensible choice.

In order for such a reduction in the state lattice's angular resolution to produce efficient (and visually pleasing) motion in narrow areas, the lattice directions and the dominant directions of the environment need to match. A simple approach for strictly orthogonal environments is to find the dominant directions of the environments via line fitting techniques, and then rotate either the maps or the lattice accordingly. A more general approach for arbitrarily oriented environments is to first compute a low-fidelity trajectory then construct a high-fidelity solution by minimizing a cost function that takes into account dominant directions in the trajectory's local environment [16].

B. Segmented Environments

Many environments exhibit strongly segmented behavior. Examples include indoor offices or homes, where individual rooms are often separated through doors and/or hallways. This kind of segmentation is not limited to indoor environments. On a larger scale, roads can be considered to be narrow and highly constrained segments, interlinking parking lots and other unstructured areas. When planning with state lattices in such constrained environments, it becomes apparent that only a *very limited number of maneuvers* are possible through these areas (note the black nodes in Fig. 3).

This property can be exploited to generate *optimal* global paths by only *locally* planning in the current segment of the unstructured area (e.g. to the beginning of the next constrained segment along the path). For the sake of clarity, an indoor space is assumed during the following elaboration.

We define a *room* to be an extended connected space of sufficient width for the robot to safely cross with dynamic obstacles. A room will by this definition never be completely blocked. A *hallway* is an elongated, narrow passage, which a robot cannot safely cross with dynamic obstacles. A *door* is a short, narrow passage, which a robot cannot safely cross with dynamic obstacles. This has the following implications:

- Humans and other dyn. obstacles passing through narrow passages block that passage completely, thereby invalidating any static heuristic. Poor heuristics lead to many expanded states close to the blockage, slowing down trajectory generation or inhibiting it completely.
- In order to leave a room, the robot potentially needs to pass through an exact set of poses.
- On-board sensors (common for mobile tasks) are generally not able to record events outside the current room.

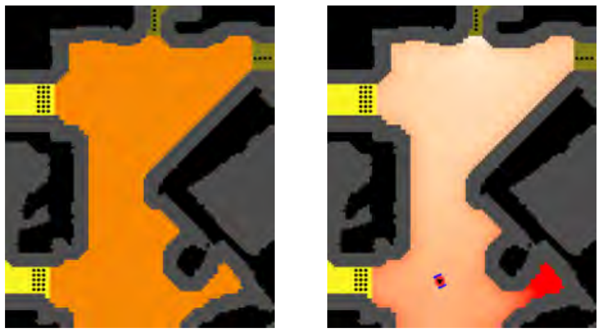
Fig. 3. Left: enlargement of the boxed area in Fig. 2-Left. Offline, a look-up table with trajectories between every pair of nodes adjacent to the same room is stored (trajectories are not shown). Right: online 2D heuristic computation. For an assumed goal towards the top, a 2D heuristic is computed. Darker reds represent higher expected cost to the top door.

In cases where the above implications hold, we propose to divide the global high-dimensional real-time navigation problem into an *offline precomputation phase*, an *online global high-dimensional heuristic generator* and a *local intra-room planner*. We can then guarantee global optimality of the resulting high-dimensional trajectory by considering the high-dimensional costs of all possible routes from the start room to the goal room, which have been pre-computed and stored in a look-up table, in addition to an online computed cost from the goal to any and all entries into the goal room and the start to any and all exits from the start room. In the following sections, each of these phases is described in more detail.

1) Segmentation: The task of separating a map into rooms, hallways and doors is non-trivial and essentially a segmentation task. One automated such approach based on range data sequences (which constitute a map of the environment) is described in [17].

2) Offline trajectory precomputation phase: When planning with a state lattice, only a very limited amount of poses are viable to lie on a trajectory inside hallways and doors. For hallways an elongated transition area should be considered, where the angle is aligned with the orientation of the walls. When transitioning through a door on the other hand, only a very small area should be considered, but the angles at which the robot may pass through the door can be relaxed.

In our approach, offline trajectories between all door-door and door-hallway combinations connected to a common room are computed in 4D and then stored in a look-up table. Additionally trajectories between the two entries of hallways are also generated. In this way, we obtain a high-dimensional estimate of the cost of traversal of any room into any other room in absence of dynamic obstacles. This 'no dynamic obstacles assumption' is justified if the robot carries all its sensors on-board, as this would prevent it from observing dynamic obstacles in remote rooms.

3) Online heuristic value generation phase: To compute a heuristic estimate of the cost to the goal, we compute the cost of a 4D trajectory from every entrance in the goal room to the goal, then use the offline look-up table to calculate the cost from each exit of the start room to the goal room. We then perform a 2D heuristic search within the start room by seeding the expansion with above obtained values to get a 2D cost estimate from every location in the start room to the goal.

4) Online intra-room planning phase: The local lattice planner then plans in the same way as in the regular 'single-area' case, except that it uses the above calculated 2D heuristic map that is only locally valid in the current room. The other major difference to 'single-area' planning is the existence of possibly (multiple) local goal areas around the various exits of the current room. Whenever a node in one of these areas is reached, the current search is successfully terminated, and the whole process eventually continues in the next room until the global goal is reached.

5) Real-time heuristic recomputation: A more accurate heuristic in best-first searches generally leads to a faster solution during the high-dimensional search (potentially by orders of magnitude, see Sec. IV). It is therefore desirable to regenerate a current and accurate 2D heuristic at every sampling instance, or at least whenever it is estimated that the improvement in overall search speed due to the updated heuristic outweighs the computational cost of updating the heuristic. Unfortunately, in very large areas, generating a 2D heuristic every cycle (using i.e. Dijkstra's search) may not be possible if real-time performance is required.

By making use of the 'segmented environments' approach, however, it is possible to recompute the 2D heuristic in the current room only, which is much faster. Both the static map and the current laser readings may serve as inputs. In this way, slow moving dynamic obstacles are included into the heuristic, which generally leads to much faster conversion in the following high-dimensional search (see Sec. IV).

Real-time 2D heuristic recomputation capability can thus be guaranteed for environments whose *largest single room* does not surpass a certain size. This allows us to provide real-time global planning in environments that are extremely large (and certainly much larger than would be possible to plan over with existing approaches).

IV. RESULTS

A. Robotic Platform

Our test system consists of a Segway RMP differential drive platform equipped with a SICK LMS 200 for laser-based localization. In conjunction with a decent map of the environment, odometry drift is of no concern.

B. Test Cases

We compare planning times for both regular *single area* 3D (x, y, θ), and 4D (x, y, θ, v) lattice planners and a 4D implementation exploiting *segmented environments* (4D segm.), with the segmentation chosen manually as depicted in Fig. 2. Each planner employs the 16-directional multi-resolution lattice presented in this paper. A suboptimality factor of ε = 15 ... 1.5 was chosen for the anytime search.

C. Long Run through the Entire Lab

The 'long run' encompasses a path of approximately 40[m], and leads from one end of the Intel Research Pittsburgh lab to the other, passing one door and several narrow hallways on its way. It therefore gives a good overview on the performance to be expected for long-range global indoor planning in the absence of dynamic obstacles.

Planner:	Heuristic[s]:	Planning[s]:	Exp. States[#]:
3D	0.11	0.04	<100
4D	0.11	2.28	5900
4D segm.	0.04	0.02	<100

Of the planners that do not exploit segmented environments, the path generated by the 3D algorithm expands by far the least number of states. This can be attributed to the 2D heuristic, which is much more accurate for the 3D case than for the 4D case, where regions of restricted maximum velocity (hallways, doors) are not well represented. Many more expanded states are therefore necessary to arrive at the same desired suboptimality guarantees (here ε = 1.5). Note the shorter time required for the 2D heuristic computation in the 4D segmented environment case. This is due to the heuristic being only locally computed in the current room (colored blue in Fig. 4.B). Real-time heuristic recomputation (i.e. at 10Hz) thus becomes feasible using this approach.

D. Short Run in a Single Room (Unoccupied)

Planner:	Heuristic[s]:	Planning[s]:	Exp. States[#]:
3D	0.11	0.01	<100
4D	0.11	0.03	<100
4D Segm.	0.06	0.03	<100

For intra-room navigation in absence of dynamic obstacles, real-time re-planning capability is observed for all three planners (planning is performed in the kitchen area of the Intel Research Pittsburgh lab, see Fig. 4.C-E). Because the start and goal locations reside in the same room, the segmented environment approach produces the same solution as the regular 4D planner. Heuristic computation time is much faster, however, as Dijkstra's search is performed in that room only.

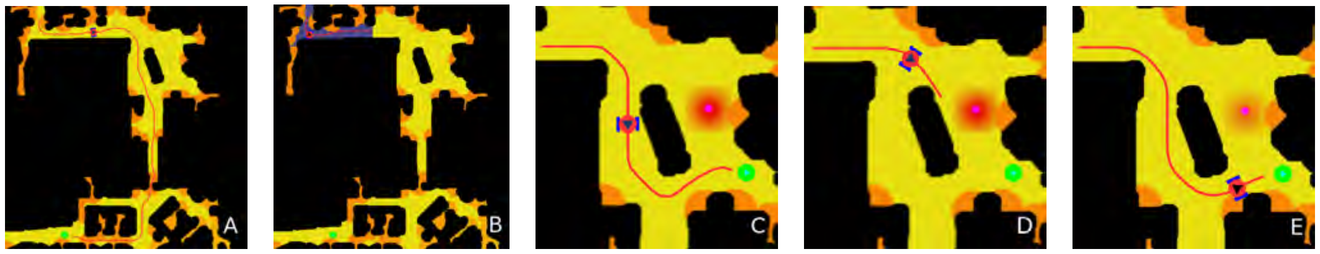

Fig. 4. Long run through the Intel Research Pittsburgh lab (Fig. 4.A-B). A: single area 4D case. B: 4D case exploiting the segmented environment. Planning is only performed in the current hallway (blue). The rest of the trajectory has been precomputed and is thus available instantly through a look-up table (not shown). Short run (Fig. 4.C-E): path favored by the 2D Dijkstra search is blocked by a dynamic obstacle (red dot). C: due to the inaccurate heuristic, the 3D planner's expansions are initially guided into the local minimum above the dynamic obstacle. Because of the low dimensionality of the search, however, the planner eventually succeeds in climbing out of it and reaching the goal. D: the single area 4D planner is also prone to the inaccurate 2D heuristic. During the deliberation time, it does not succeed in climbing out of the local minimum. E: in the segmented environment case, the Dijkstra search is recomputed in real-time, allowing for dynamic obstacles' inclusion into it. The 4D Segm. planner thus profits from an accurate 2D heuristic, which explains the fast convergence to the goal.

E. Short Run in a Single Room (Blocked Preferred Path)

Planner:	Heuristic[s]:	Planning[s]:	Exp. States[#]:
3D	0.11	5.35	19400
4D	0.11	>10.0	>25600
4D Segm.	0.06	0.08	100

If a path favored by the 2D heuristic becomes completely blocked, the reliance of the searches on the heuristic makes it difficult for them to discover alternate routes, as the heuristic constantly tries to guide the search through the blockage. The higher dimensional the search and the larger the chosen suboptimality factor ε for the anytime search, the more states close to the blockage are expanded. As the simulations show, real-time re-planning capability for all the planners relying on a static heuristic (3D, 4D) is clearly lost. It is thus important to detect such blockages, so that the robot can be stopped and the heuristic recomputed to account for it.

Wherever the environment permits, we suggest incorporating our segmented environment approach, where time often permits to continuously recompute the 2D heuristic in real-time (in the current room) based on the static map and the current laser readings. Dynamic obstacles are in this way represented in the 2D heuristic, rendering it highly accurate. This then leads to few expanded states with the 4D segm. approach, and thus to fast planning times.

V. CONCLUSIONS

In this paper, we presented a novel multi-resolution state lattice for smooth navigation in indoor environments. Furthermore, we presented a novel approach for generating fast globally optimal trajectories in constrained spaces. This approach exploits offline pre-computation to provide extremely efficient online performance and is applicable to a wide range of both indoor and outdoor navigation scenarios.

Real-time re-planning capability has been shown to be feasible in up to four dimensions, including 2D position, orientation and velocity, which are necessary to deal with dynamic obstacles. Our segmented environment approach was shown to provide significant advantage in such scenarios, where existing approaches are unable to provide solutions in the required deliberation time.

REFERENCES

[1] O. Khatib: *"Real-time obstacle avoidance for manipulators and mobile robots"*. International Journal of Robotics Research, vol. 5, no. 1, p. 90-98, 1986.
[2] R. Simmons: *"The curvature velocity method for local obstacle avoidance"*. Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), 1996.
[3] D. Fox, W. Burghard, and S. Thrun: *"The dynamic window approach to collision avoidance"*. IEEE Robotics and Automation, vol. 4, no. 1, 1997.
[4] E. Dijkstra: *"A note on two problems in connexion with graphs"*. Numerische Mathematik, 1:269-271, 1959.
[5] P. Hart, N. Nilsson, and B. Rafael: *"A formal basis for the heuristic determination of minimum cost paths"*. IEEE trans. Sys. Sci. and Cyb., 4:100-107, 1968.
[6] A. Nash, K. Daniel, S. Koenig, and A. Felner: *"Theta*: Any-Angle Path Planning on Grids"*. Proceedings of the AAAI Conference on Artificial Intelligence (AAAI), p. 1177-1183, 2007.
[7] D. Ferguson, and A. Stentz: *"Field D*: An Interpolation-Based Path Planner and Replanner"*. Advances in Telerobotics, p. 239-253, Springer Berlin, 2007.
[8] M. Likhachev, G. Gordon, and S. Thrun: *"ARA*: Anytime A* with provable bound son sub-optimality"*. Advances in Neural Information Processing Systems, MIT Press, 2003.
[9] A. Stentz: *"The focused D* Algorithm for Real-Time Replanning"*. Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), 1995.
[10] S. Koenig, and M. Likhachev: *"Improved fast replanning for robot navigation in unknown terrain"*. Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), 2002.
[11] M. Likhachev, and D. Ferguson: *"Planning Long Dynamically-Feasible Maneuvers For Autonomous Vehicles"*. Proceedings of the Robotics: Science and Systems Conference (RSS), 2008.
[12] D. Ferguson, M. Likhachev, and A. Stentz: *"A Guide to Heuristic-based Path Planning"*. International Conference on Automated Planning and Scheduling (ICAPS), 2005.
[13] D. Hsu, T. Jiang, J. Reif, and Z. Sun: *"The Bridge Test for Sampling Narrow Passages with Probabilistic Roadmap Planners"*. IEEE International Conference on Robotics and Automation, 2003.
[14] M. Pivtoraiko, and A. Kelly: *"Efficient Constrained Path Planning via Search in State Lattices"*. International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS), p.33.1, 2006.
[15] M. Rufli: *"Smooth Path Planning in Dynamic Environments"*. Master Thesis, ETH Zurich, 2008.
[16] D. Dolgov, and S. Thrun: *"Detection of Principle Directions in Unknown Environments for Autonomous Navigation"*. Proceedings of the Robotics: Science and Systems Conference (RSS), 2008.
[17] O. Martinez Mozos, and W. Burghard: *"Supervised Learning of Topological Maps using Semantic Information Extracted from Range Data"*. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2006.