The 2010 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 18-22, 2010, Taipei, Taiwan

Path Planning in 3D Environments using the Normal Distributions Transform

Todor Stoyanov, Martin Magnusson, Henrik Andreasson and Achim J. Lilienthal
Center of Applied Autonomous Sensor Systems (AASS), Örebro University, Sweden

Abstract— **Planning feasible paths in fully three-dimensional environments is a challenging problem. Application of existing algorithms typically requires the use of limited 3D representations that discard potentially useful information. This article proposes a novel approach to path planning that utilizes a full 3D representation directly: the Three-Dimensional Normal Distributions Transform (3D-NDT). The well known wavefront planner is modified to use 3D-NDT as a basis for map representation and evaluated using both indoor and outdoor data sets. The use of 3D-NDT for path planning is thus demonstrated to be a viable choice with good expressive capabilities.**

I. Introduction

The task of navigation in the context of mobile robotics can be defined in a relaxed manner as *the process of determining and maintaining a course or trajectory to a goal location* [1]. Needless to say, reliable navigation is a key requirement for autonomous robots and has been an active research area for decades. Currently, the predominant framework for navigation is built upon four components: localization, mapping, motion/path planning and path following. With the advance of six degrees of freedom Simultaneous Localization and Mapping (SLAM) algorithms accurate three-dimensional maps are becoming available online during navigation. Thus, planning in 3D environments is becoming a more feasible and an increasingly important task.

Although a lot of progress has been made in the motion planning community, applications in real world mobile systems have largely been limited to a flat floor scenario. When dealing with full 3D environments, many approaches attempt to reduce the problem dimensionality. Cost maps [2][3][4] are one popular technique that utilizes an estimate of the traversability of space, usually stored in a 2D grid. The so obtained cost maps can be coupled with a grid-based planner as in [3], or a cost-aware RRT implementation as in [2][4]. Such approaches however greatly depend on the terrain traversability estimate and may sacrifice path discovery in environments lacking a trivial 2D projection.

Implementation of full 3D-aware methods on real platforms is often hindered by the fact that modeling uncontrolled 3D environments is in general a very difficult problem. Most applications for mobile robots utilize elevation maps for terrain representation and assume that valid models are available. This however has two disadvantages — first, the uncertainty in the environment maps is usually not modeled, and second elevation grids cannot be used to represent overhanging objects and multi-story environments. Recent work has addressed these issues separately. In [5] the authors present an extension to RRT's that utilizes a particle filter framework to model uncertainty in state transitions. Further work by the same group [6] also models cost of terrain traversability to produce more efficient vehicle paths. Another approach presented in [7] utilizes a statistical framework to explicitly model the state uncertainty of a robot traversing an elevation grid. The techniques proposed in these works offer a promising research direction to cope with the inherent uncertainty in the robot state and the environment, but still are limited by the underlying elevation grid environment model.

A different approach to addressing the path planning problem is the use of a more informative environment representation scheme. A recent work from the domain of legged robot locomotion [8] uses Gaussian processes to estimate a predictive model of the terrain and handle the estimation uncertainty. The planner presented though is tailored to legged robots and can only model a single elevation level. Another approach for a legged robot [9] utilizes tri-mesh models and region segmentation algorithms to provide versatile planning capabilities. The presented approach uses a global planner for obtaining the end-to-end path and a set of local planners, suitable for the terrain type traversed. Although this approach could be migrated to a wheeled robot scenario as well, triangulation and region growing/segmentation are expensive operations and provide little utility beyond the planning task.

A novel approach to handling path planning in combined indoor and outdoor scenarios proposed in [10] utilizes a Multi-Level Surface (MLS) map. MLS is a generalization of the elevation grid terrain representation that stores a set of vertical patches in each cell, thus allowing for correct handling of overhanging objects. A benefit of this approach is that the MLS map can be obtained directly from the underlying SLAM architecture and incorporates an estimate of the uncertainty of the observed environment. The authors propose a scheme to plan paths using MLS maps [11], through a dimension projection and the extraction of a 2D binary traversability map. The so obtained occupancy map can be used as an input to a wavefront propagation algorithm [12] or another suitable two dimensional path planner. Although this approach could be used to plan paths over multi-layered terrains, the dimension projection imposes some limitations. For example, if a path is required from a specific location on any given floor of a building to the same location on another floor, a 2D map cannot correctly represent the overlapping sections of the environment.

This article proposes to address the limitations of current path planners by using the Three-Dimensional Normal Distributions Transform (3D-NDT) for spatial representation. 3D-NDT provides a compact, yet expressive environment description and is well suited for point set registration and mapping. Thus, to achieve an integrated approach to the autonomous navigation problem, it is advantageous to explore the use of 3D-NDT as a path planning data structure. The well known wavefront path planning algorithm is extended to operate on 3D-NDT spatial representations, avoiding dimension projection and possible loss of significant information. The next section proceeds with a description of 3D-NDT and a summary of its applications. Section III describes the proposed path planner. Finally Section IV demonstrates the feasibility of the approach on real-world data sets.

II. 3D-NDT

The Normal Distributions Transform was originally developed for 2D laser scan registration [13]. The central idea is to represent the observed range points as a set of Gaussian probability distributions. NDT was later extended to three dimensions [14] and applied to the domains of 3D scan matching and loop detection [15], as well as change detection [16]. One of the key advantages of 3D-NDT is the fact that it forms a piecewise smooth spatial representation, resulting in the existence of analytic derivatives. Thus, 3D-NDT can be coupled with standard optimization methods to produce state of the art registration of 3D point clouds. The correctly registered 3D-NDT scans can then be stacked together, possibly using a global network optimization scheme, to form a map of the explored environment.

The planner proposed in the next section assumes that a 3D-NDT map is already available. Nevertheless, as several possibilities exist for computing and storing NDT cells, the remainder of this section describes the procedure used in our implementation. Given a set of N sampled points $P = \{(px_i, py_i, pz_i)|i = 0...N\}$, the first step of the algorithm constructs an OctTree representation of P. The use of an OctTree for storage of 3D-NDT cells in general is not necessarily an optimal choice, but was deemed a good candidate for the needs of the proposed path planner. OctTree leafs usually have pre-set sizes, typically in an attempt to obtain well fitting Gaussian distributions in each cell. Due to differences in the sparsity and size of the data sets processed, the choice of an optimal leaf size can vary substantially. It is therefore advantageous to attempt to estimate a good leaf size adaptively, based on the observed points.

The 3D-NDT implementation used initially sets a conservative pre-set leaf size (a cube with sides of four meters). Next, a Gaussian distribution with mean μ and covariance C is estimated for each leaf, using the respective point sets $P_{leaf} = \{(px_i, py_i, pz_i)|i = 0...l\}$:

$$\mu = \frac{1}{l}\sum_{i=0}^{i=l}\begin{bmatrix} px_i \\ py_i \\ pz_i \end{bmatrix} \quad (1)$$

$$M = \begin{bmatrix} px_0 - \mu_x & .. & px_l - \mu_x \\ py_0 - \mu_y & .. & py_l - \mu_y \\ pz_0 - \mu_z & .. & pz_l - \mu_z \end{bmatrix} \quad (2)$$

$$C = \frac{1}{l-1}MM^T \quad (3)$$

At this point some of the cells might hold points from different objects, resulting in a distribution that does not correctly represent the environment. This effect is due to the inherent unimodality of Gaussian PDFs that is not necessarily reflected in the observations. Unfortunately, statistical tests for unimodality of multivariate distributions are quite complicated and time consuming. Thus, a heuristic approach was used to determine if leaf cells should be split to obtain better fitting approximations. The test used computes a residual $\epsilon_i = p_i - \mu$ for each observation p_i and an overall residual variance σ_ϵ^2. A measure of the model fitness is obtained:

$$fit = \frac{\sum_{i=0}^{N}\epsilon_i^2}{\sigma_\epsilon^2} \quad (4)$$

The proposed fitness criterion penalizes models with a high frequency of outliers, as these result in disproportionately large residuals. A smaller fitness value indicates a better coverage of the observed points and no need to split the cell. In the proposed implementation, an empirically set threshold of 1000 is used and all cells are further split until satisfying the fitness criterion or reaching a minimal cell size (see Table II in Section IV for a summary of all parameters). The results obtained with this ad-hoc approach were satisfactory. Further improving the approximation accuracy of the 3D-NDT representation was thus left as a future research direction, as it is not the focus of this work.

After estimating the 3D-NDT map the original points are no longer necessary and can be disposed of. Thus, just nine values are used to represent the local surface shape in a cell — three for the mean value and six for the symmetric covariance matrix. This greatly reduces the storage requirements for a map and ensures that the size depends on the explored area and not on the number of observed points. It is important to note that after representing the point set as a set of Gaussian distributions, the only information that the path planner can utilize is what is available from the 3D-NDT data structure — namely the means and covariance matrices for each cell. An example of a point cloud and its 3D-NDT representation is shown in Fig. 1.

III. Using the 3D-NDT for Path Planning

This section describes the proposed extension of the grid-based wavefront propagation algorithm to a 3D-NDT based map. It is usually not computationally feasible to use grid-based path planning methods in higher-dimensional problems. Note however that though the operational environment of the robot is fully 3D, the dynamics of the vehicle constrain it to locally move on a plane, possibly inclined up to a maximal vehicle specific roll and pitch. Thus, there is no need to propagate the wavefronts in a vertical direction or towards cells belonging to non-traversable terrain (walls in an indoor environment for example). The wavefront algorithm is intuitive, easy to implement and has guaranteed convergence, thus making it a good choice for a base path planner on 3D-NDT maps. The proposed modified planner is introduced in Algorithm 1 and further discussed in the following paragraphs.

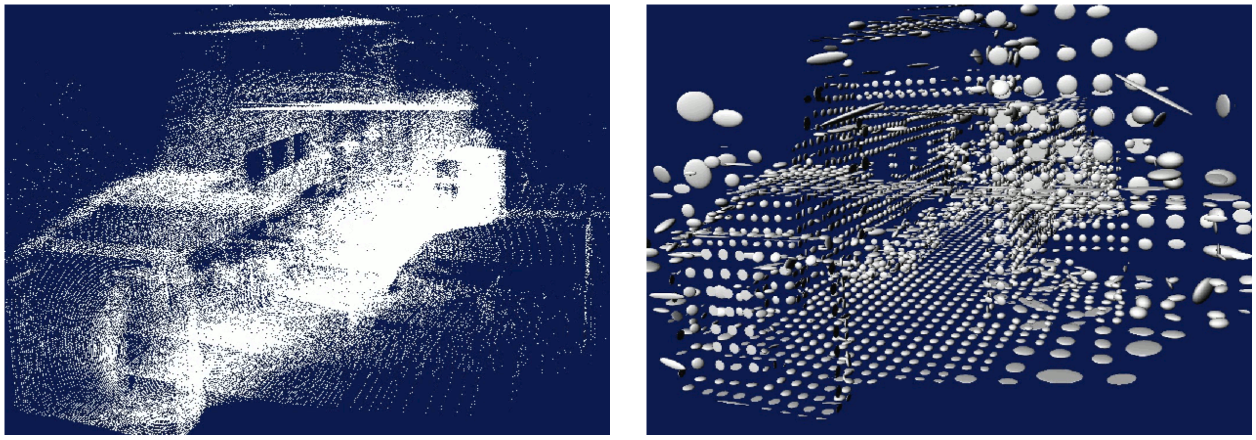

Fig. 1: Left: A point cloud acquired with a 3D laser in an indoor hallway environment. Right: 3D-NDT representation of the same point cloud. Ellipsoids are used to visualize the Gaussian distributions in each cell. All ellipsoids are centered at the distribution means and orientated and scaled according to the respective covariance matrices

Algorithm 1: The 3D-NDT wavefront path planner

```
1: Initialize active cell list $Q$
2: $\forall q \in \mathcal{C} : q.cost = \infty$
3: $q_{goal}.cost = 0$
4: $Q \leftarrow q_{goal}$
5: while $Q$ not empty do
6:     $q_{cur} \leftarrow Q.pop()$
7:     if $CollisionCheck(q_{cur})$ then
8:         $Q_{next} \leftarrow AccessibleNeighbors(q_{cur})$
9:         if $Q_{next}.cost > q_{cur}.cost$ then
10:            $Q_{next}.cost = q_{cur}.cost + cost(Q_{next}, q_{cur})$
11:        $Q.push(Q_{next})$
```

The algorithm presented is very similar to the 2D version of wavefront propagation, but has two important distinctions that utilize the 3D-NDT representation. The first modified component is the $CollisionCheck$ subroutine. While in a 2D grid based approach this routine is a simple check of the binary occupancy value of the cell, the modified version is more involved. The second modification is in the $AccessibleNeighbors$ routine, which has also been changed to accommodate the specific constraints of a vehicle moving in a 3D environment. Finally, in order to handle the non-uniformity of the 3D-NDT grid, the cost function propagated $cost(Q_{next}, q_{cur})$ is proportionate to the Euclidean distance between the cells considered. Throughout this section, typical path planning notation is used — namely $\mathcal{C} : \{q\}$ is used to denote the *configuration space* of the robot.

A. State collision detection (CollisionCheck)

As the size of the 3D-NDT cells is generally not correlated to the size of the robot, an accurate collision check routine should evaluate a local neighborhood containing the tested configuration. Thus, a nearest neighbor search for all cells within a sphere with a double of the robot radius is performed in the OctTree. Using this immediate robot neighborhood, the routine in Algorithm 2 evaluates the safety of the configuration state q. The cells in the immediate neighborhood N are split in two classes — possible support and possible colliding cells. Support cells cover space that could potentially be

Algorithm 2: $CollisionCheck$

```
1: Input $q \leftarrow$ configuration state to be evaluated
2: $N \leftarrow$ all OctTree cells within 2 x RobotRadius
3: for $\forall n \in N$ do
4:     if $Angle(n, q) < maxPitch$ then
5:         $pSupportCells \leftarrow n$
6:     else
7:         $pCollisionCells \leftarrow n$
8: if $\exists n_i \in pSupportCells$ : not horizontal or inclined then
9:     return Collision
10: if $\exists n_i \in pCollisionCells : Collides(q, n_i)$ then
11:    return Collision
12: return No Collision
```

traversed by the vehicle, while offering a stable kinematic support. On the other hand, possible collision cells cannot be traversed and belong to obstacles that can hinder the robot. The classification is performed by evaluating the angle that the vector $\mu_{current} - \mu_{neighbor}$ makes with the horizontal plane. If this value is larger then the maximum vehicle pitch the neighbor is regarded as a possible collision. Once the cells are classified, two further conditions need to be satisfied — namely that $pSupportCells$ indeed provide a stable support for the robot and that the $pCollisionCells$ do not actually collide with the robot body.

In order to check the stability of the support cells, first the traversability of each cell in the set is assessed. As proposed in [15], a threshold on the smallest eigenvalue of the covariance matrix is used to determine the roughness of the terrain. The inclination of the Gaussian in the NDT cell is computed and compared to the maximum allowed pitch of the vehicle. In practice the roughness and inclination are precomputed for the entire map and the cells are classified as rough, horizontal, inclined or vertical planar cells (Fig. 2). Thus, this step amounts to querying the class of each cell.

Finally, the cells that do not belong to the robot support are evaluated for collision with the robot sphere. This routine (line 10 in Algorithm 2) is performed in two steps. First, a

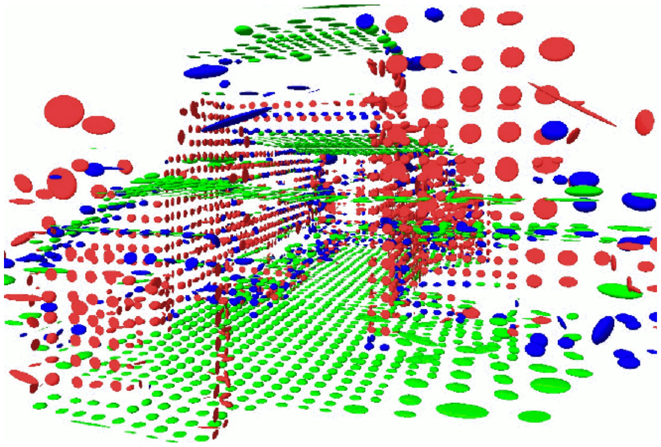

Fig. 2: Classified cells for the scan shown in Fig. 1. In a colored print dark blue signifies a sharply inclined cell, red is used for vertical flat surfaces, green is used for flat horizontal cells and cyan for rough cells (not in this image).

fast check is performed, testing if the mean to mean distance between the current and neighboring cell is bigger then the robot radius — $|\mu_{diff}| = |\mu_{current} - \mu_{neighbor}| > r$. Even if this condition is satisfied, it is still possible that a collision might occur. A fast conservative check for a collision is performed using a re-scaled mean to mean distance μ_{sc}.

$$\mu_{sc} = \frac{|\mu_{diff}| - r}{|\mu_{diff}|} \mu_{diff} \tag{5}$$

The vector μ_{sc} estimates the distance between the mean of the neighboring cell and the edge of the robot sphere. A Mahalanobis distance based on the neighbor's covariance matrix C_{neigh} thus provides a measure of the likelihood that the endpoint of this vector belongs to the neighbor's distribution. Thus, the final collision check is performed by thresholding $\mu_{sc}^T C_{neigh} \mu_{sc} < 1$.

B. Finding accessible neighbors

The second modification to the 2D wavefront algorithm is in the $AccessibleNeighbors$ routine. In the original wavefront algorithm this call returns the free cells with adjacent indexes in the occupancy grid map. The logic behind this routine is kept, but the implementation is changed to accommodate for the 3D-NDT representation used. In practice, the support cells extracted from the immediate neighborhood evaluated in $CollisionCheck$ are cached and used as the input for this routine. To ensure that it is possible to move from the current configuration q_{cur} to a cell in the support set, a further test is performed using the $TransitionPossible$ routine (Algorithm 3). Cells are deemed non-accessible if the mean-to-mean distance is more than the sum of the diagonals of the cells (not direct neighbors) or if there is a large difference in the orientation of the local surface ellipsoids.

C. Planning the path

Upon termination of the 3D-NDT Wavefront algorithm, all cells from which the goal is reachable are assigned a cost value. In case the cost of the start location is finite, a path to the goal exists and can be generated by following the cost function gradient, starting from q_{start} and iteratively checking for an accessible neighbor with lower cost.

Algorithm 3: $TransitionPossible$

```
1: Input $q_{cur}$ and $q_{neigh}$ state transition to be tested
2: $s \leftarrow$ sum of half diagonals of the cells
3: if $|\mu_{cur} - \mu_{neigh}| > s$ then
4:     return false
5: $v_c \leftarrow$ eigenvector with smallest eigenvalue for $C_{cur}$
6: $v_n \leftarrow$ eigenvector with smallest eigenvalue for $C_{neigh}$
7: if $Angle(v_c, v_n) > maxPitch$ then
8:     return false
9: return true
```

IV. Results

In order to demonstrate the validity and utility of the proposed path planning algorithm, test runs were performed in four different environments. The data sets used were collected by means of an actuated two dimensional range scanner (SICK LMS), using different mobile platforms. The logged odometry pose data and point sets were used as input to a 3D-NDT based scan matcher, as in [15]. The registered scans were then merged and stored for use in several path planning queries. The point sets used are shown in the left column of Fig. 3, while the right column shows the reachable 3D-NDT cells, produced by the planner. The remainder of this section discusses the features of each environment and the performance of the proposed algorithm.

The first environment tested is an example of a typical indoor scenario — a robot moving through a hallway with flat floors. Even in such classical situations, there are a number of challenges that are better faced with a 3D planning strategy. One such example occurs when planning close to tables or chairs, that normally have thin legs and might not be visible in a 2D laser scan. The proposed approach handles these areas elegantly thanks to the 3D collision evaluation. As the potential map reflects, the areas near the top wall of the hallway are correctly considered unreachable, due to the presence of tables and chairs.

The second and third test environments consist of service tunnels in a mine. Although the floors are mostly flat, there are a number of rough floor areas, as well as ditches, that produce holes in the point clouds. The planning task is further complicated by the presence of other service vehicles, that occupy some of the otherwise traversable areas of the data sets. The potential maps obtained for these data sets are also sound — rough and unobserved areas of the corridors are not present in the final maps and thus avoided when planning paths. Note that in the Kvarntorp data set, due to a ditch there is no connectivity in the upper left corner of the map.

The last test point set used is from an outdoor asphalt processing site. The environment consists of uneven and sloping terrain, including several piles of gravel and asphalt. Paths were planned in this map using a vehicle model that allows for a maximum pitch of up to 0.2 rad, which is

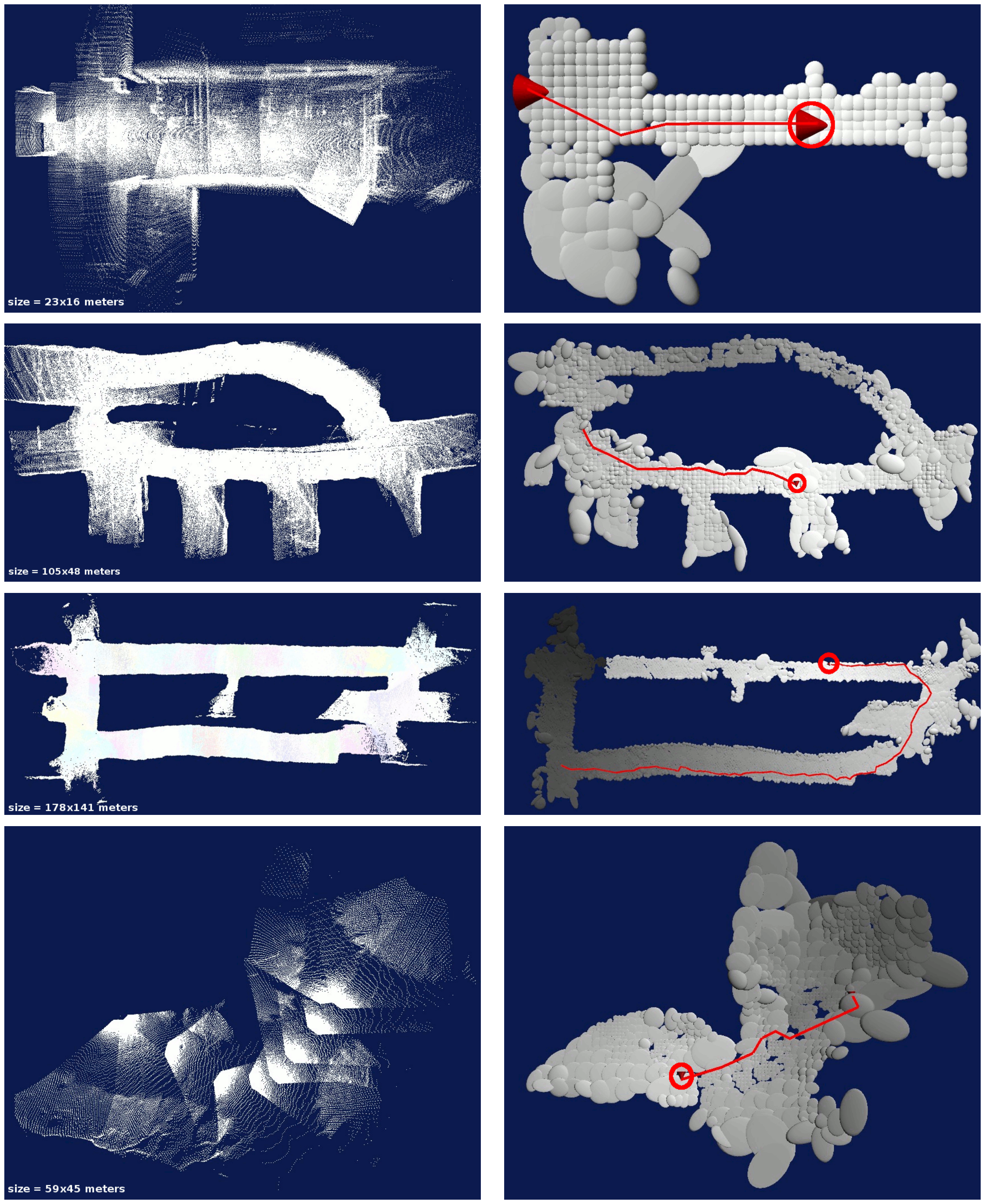

Fig. 3: Left: Point clouds from four test environments. From top to bottom data sets are labeled ”hallway”, ”mine”, ”Kvarntorp” and ”piles”. For description of the environments, check Section IV. Right: Wavefront propagation maps for the respective environments. Ellipsoids are scaled (x3) for display purposes. Colors show proximity to the goal. Planned paths are shown as red lines, with start and goal locations marked by red cones. Only reachable cells shown.

reflected in the fact that the the pile foundations are still considered traversable in the output map.

For each of these four environments, a goal location that coincides with a free state was selected. Start locations, also considered collision free were selected for each map and used as an input to the proposed algorithm. Since ground truth terrain traversability was not available for the collected data sets, the feasibility of the produced paths was evaluated only empirically. The quality of the paths, as demonstrated in Fig. 3, exhibits the typical characteristics of a gradient-following approach. The paths are geometrically feasible, but are not necessarily optimal in minimizing jitter or vehicle travel time. Note that the presented approach is indeed a pure *path planner* and as such doesn’t take into consideration the dynamics of the robot. Another feature of the data sets that has to be mentioned is that although some of the environments contain tunnels, truly overhanging structures and multi-layered traversable terrain are not present.

Data Set	Points	NDT Cells	Reachable Cells	NDT Time	Planning Time
Hallway	636k	4 423	282	12.5s	0.2s
Mine	863k	21 460	1 635	16.6s	1.8s
Kvarntorp	1 528k	31 269	4 120	26.5s	3.1s
Piles	183k	1 361	782	3.3s	0.6s

TABLE I: Environment complexity and runtime statistics

The performance of the planner is influenced by the metaparameters in Table II. The maximum and minimum cell size and model fitness affect the quality of the 3D-NDT map and thus also the path quality. Notably, increasing the minimum cell size or lowering the fitness threshold would speed up planning, but could also induce errors in modeling small details like narrow passages. The Mahalanobis and eigenvalue thresholds change the behavior of the collision checker and the discrimination of safe states. Increasing the parameters allows for larger clearance from obstacles and rougher horizontal cells respectively. The last two parameters model the physical characteristics (width and tipping pitch angle) of the robot executing the path.

The complexity of the environments tested and the average path planning times (using an Intel Core2 Duo CPU at 3 GHz) are shown in Table I. Several general observations can be made, based on the tests performed. The first notable result is that the path planning time (column 6) is dominated by the time necessary for building the 3D-NDT representation of the environment (column 5). This is not surprising, as the planner utilizes information available in the NDT map to speed up the search. It is important to note that in an online navigation scenario, the 3D-NDT map would be available by default as it is an integral part of the mapping system. A second observation to note is that the planning time depends more on the number of reachable cells (column 4) then on the number of cells in the map (column 3). This relation can be explained by the fact that searching for a cell in the OctTree has an expected computational cost of $\mathcal{O}(log(N))$. This operation is performed in a manner proportionate to the number of expanded cells K, as each cell has a finite number of neighbors. Thus, the complexity of the path planner can be approximated as $\mathcal{O}(Klog(N))$, which in the presented examples is dominated by the number of reachable cells.

V. Conclusion

This article proposed a novel approach to path planning for wheeled mobile robots in semi-structured 3D environments. The use of 3D-NDT as an underlying map representation allows for an expressive and computationally effective spatial modeling. A well known 2D path planning algorithm is extended to operate on a 3D-NDT map, thus removing the need for expensive terrain modeling and segmentation operations. The proposed approach was demonstrated to produce feasible paths in four different test environments, with varying characteristics. The use of 3D-NDT as a spatial data structure was shown to be beneficial for reducing the complexity of the proposed path planner, removing the necessity of projecting into two dimensions and the possible loss of path diversity.

Max Cell Size	Min Cell Size	Model Fitness Thresh	Mahal-anobis Thresh	Eigenval. Roughness Thresh	Robot Radius	Max Pitch
4m	0.4m	1000	1	0.1	1m	0.2rad

TABLE II: Meta-parameters and their values

The algorithm presented is a grid planner and thus suffers from the limitations of this class of approaches. Therefore, an interesting direction for future work is the extension of a sampling based planner to 3D-NDT maps. Another direction to be explored is the comparison between paths produced by the proposed algorithm and by existing approaches. Path quality estimation in real world 3D environments with no ground truth is still an open question and should be addressed to facilitate an objective comparison.

Acknowledgments

This work is partially sponsored by the All-4-eHAM project (http://aass.oru.se/Research/Learning/all4eham/index.html)

References

[1] M. Franz and H. Mallot, "Biomimetic robot navigation," *Robotics and Autonomous Systems*, vol. 30, pp. 133–153, 2000.

[2] J. Lee, C. Pippin, and T. Balch, "Cost based planning with RRT in outdoor environments," in *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems. IROS 2008.*

[3] B. Gerkey and K. Konolige, "Planning and control in unstructured terrain," in *Proc. of the Workshop on Planning with Cost Maps, IEEE Int. Conf. on Robotics and Automation*, 2008.

[4] D. Ferguson and M. Likhachev, "Efficiently Using Cost Maps For Planning Complex Maneuvers," in *Proc. of the Workshop on Planning with Cost Maps, IEEE Int. Conf. on Robotics and Automation*, 2008.

[5] N. Melchior and R. Simmons, "Particle RRT for path planning with uncertainty," in *Proc. of IEEE Int. Conf. on Robotics and Automation, ICRA 2007.*

[6] J.-Y. Kwak, M. Pivtoraiko, and R. Simmons, "Combining cost and reliability for rough terrain navigation," in *9th Int. Symp. on Artificial Intelligence, Robotics and Automation in Space (iSAIRAS'08).*

[7] G. Kewlani, G. Ishigami, and K. Iagnemma, "Stochastic mobility-based path planning in uncertain environments," in *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems. IROS 2009.*

[8] C. Plagemann, S. Mischke, S. Prentice, K. Kersting, N. Roy, and W. Burgard, "Learning predictive terrain models for legged robot locomotion," in *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems. IROS 2008.*

[9] R. B. Rusu, A. Sundaresan, B. Morisset, K. Hauser, M. Agrawal, J.-C. Latombe, and M. Beetz, "Leaving Flatland: Efficient Real-Time 3D Navigation," *Journal of Field Robotics (JFR)*, 2009.

[10] R. Triebel, P. Pfaff, and W. Burgard, "Multi-level surface maps for outdoor terrain mapping and loop closing," in *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems. IROS 2006.*

[11] P. Pfaff, R. Kümmerle, D. Joho, C. Stachniss, R. Triebel, and W. Burgard, "Navigation in Combined Outdoor and Indoor Environments using Multi-Level Surface Maps," in *In Workshop on Safe Navigation in Open and Dynamic Environments at IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS 2007.*

[12] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006, available at http://planning.cs.uiuc.edu/.

[13] P. Biber and W. Straßer, "The normal distributions transform: A new approach to laser scan matching," in *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems. IROS 2003.*

[14] M. Magnusson, T. Duckett, and A. J. Lilienthal, "3d scan registration for autonomous mining vehicles," *Journal of Field Robotics*, vol. 24, no. 10, pp. 803–827, Oct 24 2007.

[15] M. Magnusson, "The three-dimensional normal-distributions transform — an efficient representation for registration, surface analysis, and loop detection," Ph.D. dissertation, Örebro University, Dec. 2009.

[16] H. Andreasson, M. Magnusson, and A. J. Lilienthal, "Has something changed here? autonomous difference detection for security patrol robots," in *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems. IROS 2007.*