Distance-Aware Dynamically Weighted Roadmaps for Motion Planning in Unknown Environments

Adrian Knobloch, Nikolaus Vahrenkamp, Mirko Wächter and Tamim Asfour

***Abstract*—The paper presents and evaluates a Distance Aware Dynamic Roadmap (DA-DRM) algorithm as an extension of the Dynamic Roadmap (DRM) approach. In contrast to previous work, the algorithm is capable of planning collision-free trajectories while considering the distance to obstacles, even in unknown environments which are perceived by the robot's depth camera system. The algorithm makes use of a voxel distance grid which is updated based on perceptual information acquired from the robot's perception system. The distance information is considered as a cost factor during the roadmap search and it is considered in a post-processing step that is used for trajectory smoothing. We evaluate the DA-DRM algorithm in simulation and in a real-world experiments with the humanoid robot ARMAR-III. In addition, we compare our algorithm against the DRM and the RRT-Connect algorithm. The results demonstrate the performance of our algorithm in terms of keeping a safety distance to obstacles, trajectory smoothness as well as the ability to generate solutions in narrow free space.**

***Index Terms*—Motion and Path Planning Collision Avoidance**

I. INTRODUCTION

THE efficient generation of collision-free trajectories for a robot in partly known or even unknown environments is a challenging task. In contrast to motion planning approaches, which assume a perfect world knowledge, approaches for real world applications should be able to consider unknown obstacles and dynamically changing environments. Hence, planning algorithms which are capable of integrating perceptual input are indispensable in open-ended environments. When considering human-robot interaction scenarios, the predictability of the robot's movement is an additional requirement that leads to higher confidence and acceptance by humans.

Thus, a motion planning algorithm should be aware of the distance to obstacles to allow generating trajectories with a safety distance to obstacles to avoid collisions which may occur due to sensor and actor inaccuracies or to unreliable hand-eye calibration in grasping and manipulation tasks.

We present a motion planning algorithm capable of generating collision-free trajectories based on perceived depth images without the need of computer vision algorithms for object detection and localization. In addition, the planned motions of the Distance Aware Dynamic Roadmap (DA-DRM) algorithm considers the obstacle distance in order to avoid movements that result in unnecessary low distances to the environment.

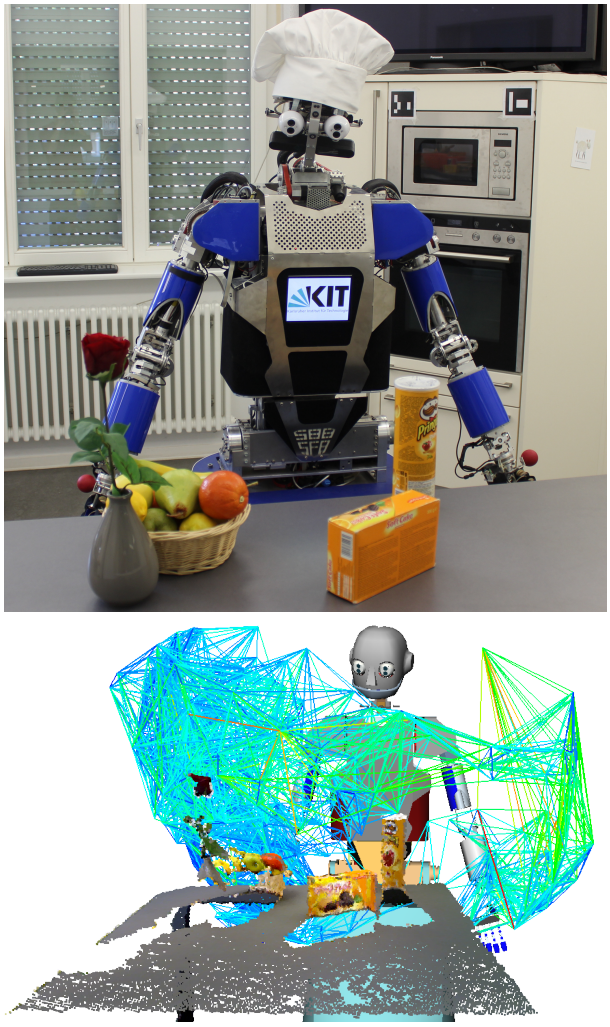


Fig. 1. The scene (top) is used to generate the point cloud shown in the bottom using the depth camera of the ARMAR-III. The adapted roadmap graph is shown in workspace (bottom). The edge costs which are associated with obstacle distance are shown in different colors (low: blue; high: red).

II. RELATED WORK

Motion planning has been a highly addressed research topic in robotics in the last decades and a wide variety of algorithms have been developed. The most common types of planning

Manuscript received: 09/10/2017; Revised 12/16/2017; Accepted 01/15/2018.

This paper was recommended for publication by Editor Nancy Amato upon evaluation of the Associate Editor and Reviewers' comments. The research leading to these results has received funding from the European Unions Horizon 2020 Research and Innovation programme under grant agreement No 643950 (SecondHands).

The authors are with the High Performance Humanoid Technologies (H^2T) Lab, Institute for Anthropomatics and Robotics (IAR), Karlsruhe Institute of Technology (KIT), Germany adrian.knobloch@student.kit.edu, {vahrenkamp,waechter,asfour}@kit.edu

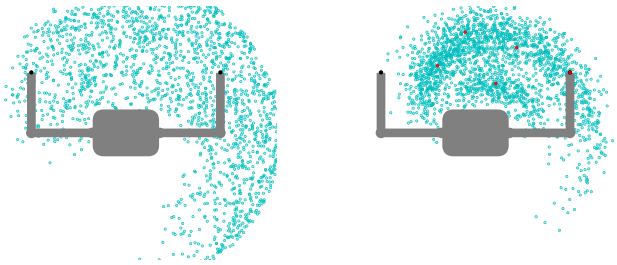
Digital Object Identifier (DOI): see top of this page.

algorithms are Rapidly-exploring Random Tree (RRT)- and Probabilistic Roadmap (PRM)-based approaches. The RRT is single-query motion planning algorithm which was developed by Kuffner and LaValle [1] and served as the basis for many extensions and variations. RRT-based approaches have the advantage that no preprocessing is needed and that they can be applied in high-dimensional configuration spaces, and in various domains. In contrast to such single-query methods, PRM-based approaches are multi-query algorithms. These algorithms require pre-calculated data to plan a collision-free trajectory, but due to the preprocessed data, PRMs can be queried very efficiently. The standard PRM approach [2] needs collision informations during the preprocessing stage in order to generate the roadmap. Since these informations cannot always be provided in real-world applications, several algorithms have been developed to address this issue. One possibility is to plan a trajectory for the static part of the environment using a PRM and then adapt this trajectory to the dynamic parts [3]. Another approach, which even works for completely unknown environments is the Dynamic Roadmap (DRM) algorithm ([4], [5]). As shown by Kallmann and Mataric [6], DRMs can be used with a point cloud representation of the actual scene. This DRM algorithm has been used in various works, in particular to improve the efficiency of the approach to make it applicable for online usage ([7], [8], [9], [10], [11], [12]). A critical issue of RRT and PRM based approaches is the quality of the planned trajectories in terms of smoothness and length. Several methods have been proposed to deal with these issues. For PRM-based algorithms, the quality of the trajectories could be improved by improving the roadmap itself ([13], [14]). Other approaches optimize the trajectory in an post-processing step. This can be achieved by randomly creating shortcuts on the path [15] with elastic band algorithms [16], or the CHOMP approach [17].

III. DISTANCE-AWARE DYNAMIC ROADMAPS

We present a distance-aware dynamic roadmap algorithm, the DA-DRM, which is based on the DRM algorithm [4] and which is is able to directly work with point clouds. In the following, we briefly describe the basic DRM approach and our extensions which result in the new DA-DRM algorithm. Finally, we present a post-processing step for the DA-DRM algorithm, which smooths the generated trajectories while considering the obstacle distance information generated during planning.

The DRM algorithm is divided into two steps: an offline roadmap generation step and the online trajectory generation step. The roadmap generation consists of the following steps: 1) configuration sampling, where we use preferred sampling around configurations in the main working area as extension, 2) generation of the DRM graph using k-nearest neighbour search, and 3) establishment of the mapping between the voxelized workspace and the nodes and edges of the graph Using the resulting roadmap, the online trajectory generation is achieved by the following steps: 1) Calculation of the occupied workspace voxels, where additionally voxels next to the occupied voxels and their distance to the blocked part are calculated, 2) removing of blocked roadmap edges and additionally updating the weight of edges near to the blocked part of the environment, and 3) trajectory generation with A* search on the weighted graph



(a) Coverage of the workspace for configurations sampled using a uniform distribution.

(b) Coverage of the workspace for configurations sampled using a seeded normal distribution.

Fig. 2. The roadmap coverage of a 3-joint robot. Each point represents the TCP position of one configuration, where the sampled configurations are blue and the seed configurations are red.

A. Structure and Generation of the DRM

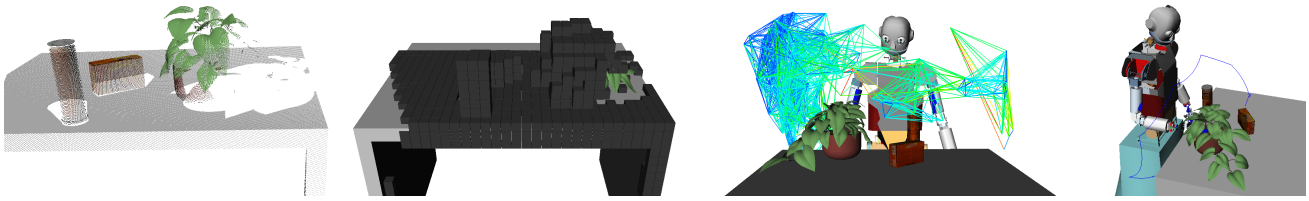
Since DRMs are based on PRMs, a weighted graph $G = (V, E)$ is needed, where V consists of joint configurations c_i of the robot's configuration space (C-space). A configuration is an ordered set of joint values $c_i = (c_i^0, c_i^1, \ldots, c_i^N)$, where each value corresponds to a specific joint and N the number of degrees of freedom. Guided by the observation that the whole workspace of the robot is not always of interest for many tasks (e.g. the area behind the robot), we bias the roadmap generation to be around some initial seeding points, which are selected manually in workspace areas of interest. For this purpose, a normal distribution $\mathcal{N}(\mu, \sigma^2)$ is used to sample the workspace around initially chosen seed values $s_0, s_1, \ldots, s_k$:

$$c_i^j = \mathcal{N}(s_{i \bmod k}^j, \sigma^2)$$

As shown in Fig. 2, this results in less configurations behind the robot. The edges of the graph are generated by choosing the k-nearest configurations for each configuration c_j. Such an edge $e_i = \{c_j, c_k\} \in E$ describes a trajectory from one configuration to the another. This trajectory is the linear interpolation between the two configurations. Since the interpolation is symmetric, the edges are assumed to be undirected. This also increases the amount of possible paths through the roadmap, but requires the weight to be equal for each edge and the reverted edge. The weight of an edge is the distance between the two configurations. To this end, the extended workspace distance metric d^{wm} presented by Kunz et al. [18] is used:

$$d^{wm}(c_i, c_j) = \sqrt{d^w(c_i, m)^2 + d^w(m, c_j)^2}\ ,$$

where the configuration m is the midpoint on the linear interpolation of $c_i \to c_j$. This metric is further evaluated against C-space metrics d^c and workspace metrics d^w in "Distance metrics for path planning with dynamic roadmaps" [13]. Voelz and Graichen come to the conclusion that the distance metric



(a) (b) (c) (d)

Fig. 3. The DA-DRM algorithm consists of three stages: 3(a) → 3(b): Calculation of the occupied workspace voxels using point cloud data; 3(b) → 3(c): Adapt the DA-DRM graph to the current state of the environment; 3(c) → 3(d): Search the cheapest path through the graph

d^{wm} provides better results than the d^c and d^w distance metrics.

A DRM also contains a voxel mapping ϕ: $W_v \rightarrow V \cup E$. The voxelized robot workspace W_v is a three dimensional grid containing $x \times y \times z$ voxels, each with fixed dimensions, where x, y and z are limited by the reachable area of the robot. The function ϕ saves the workspace to c-space mapping (W-C mapping) and returns for each voxel $v \in W_v$ all nodes and edges, which collide with it.

B. Trajectory generation

The planning process for DRMs is divided into three planning steps, which are visualized in Fig. 3. In the first step, the point cloud, captured by a depth camera, is processed. Therefore, each point is mapped to the corresponding voxel of the robot's voxelized workspace W_v. Each voxel of this voxelized workspace is then, in the second step, mapped to a list of nodes and edges using the W-C mapping function ϕ. These nodes and edges are disabled to adapt the graph to the environment. The third step is the actual planning step, where an $A^\star$ algorithm is used to generate the trajectory.

For the DA-DRM algorithm, we extend the second step. In addition to the disabling of all blocked nodes and edges, the weight $w(e_i)$ of all edges $e_i \in \phi(A)$, which are close to the blocked part of the environment, is adjusted.

Therefore, the adjacent voxel $a \in A \subset W_v$ of all blocked voxel B are processed and updated with their distance to the blocked part:

$$d_v(a) = min_{b \in B}(voxelBetween(a,b) * voxelEdgeLength),$$

where $voxelBetween(a,b)$ describes the number of voxels between a and b (excluding a and b). As shown in Fig. 4, only the next r voxel are updated for each blocked voxel, while r has to be large enough to fully cover the safety distance d_{min}:

$$(r-1) * voxelEdgeLength \geq d_{min}$$

This distance value d_v is then used to update the weight of the corresponding edges $e_i \in \phi(A)$, which are determined by the W-C mapping. To do so, for each edge the minimal distance $d_v^{min}(e) = min(d_v(\phi^{-1}(e)))$ of all corresponding voxels is considered. The weight is computed by taking into account a minimum safety distance d_{min} and a penalty factor $p > 1$:

$$w_{new}(e_i) = w(e_i) * p^{d(e_i)}, \text{ with}$$

$$d(e_i) = \begin{cases} 0 & \text{if } d_v^{min}(e_i) > d_{min} \\ \frac{d_{min} - d_v^{min}(e_i)}{voxelEdgeLength} & \text{otherwise} \end{cases}$$

The factor p specifies which edges should be selected during the trajectory creation. While a high value for p blocks the safety area completely, a lower value allows the cutting of edges for smoother trajectories.

This results in a graph, in which all edges, which are close to objects are more expensive than edges, which are far away from the objects, as shown in Fig. 1. As a result, the roadmap edges which haven't been updated are favored during planning. If this is not possible, the edges, which are as far as possible away from the blocked parts of the environment are used. Hence, the resulting trajectory follows the safety distance if possible.

C. Post Processing

A trajectory generated by a DRM or PRM usually includes unnecessary movements, sharp edges, and clipped transitions from one edge to the other. This is a result of the creation process of the graph, where only short edges are added in the basic algorithm. To counteract these artifacts, the trajectory should consist of long smooth parts. Similar to the approach proposed in Geraerts and Overmars [14], such smoothing can be achieved by adding additional long edges to the graph. However, this leads to a significant increase of the graph

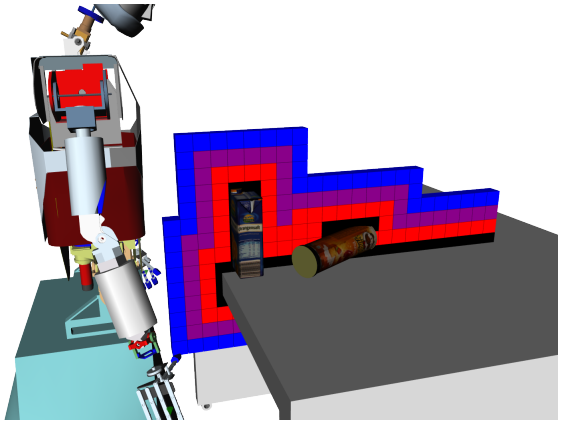


Fig. 4. All adjacent voxels (colored based on their distance) of the blocked voxels (black) are updated according to their distance to the blocked part. For a voxel size of 40 mm × 40 mm × 40 mm, the voxel distances are: red: ≥ 0 mm, violet: ≥ 40 mm, blue: ≥ 80 mm

size. Therefore, the shortcuts are only created during a post processing step by which some nodes of the trajectory are removed and the adjacent nodes are directly connected.

To find all nodes that can be removed without violating the safety distance or creating a collision, the distance grid represented by the function d_v is used. Using this grid and the W-C mapping ϕ, each node n can be flagged with its distance to the objects in the environment. If none of the voxels $a \in \phi^{-1}(n)$ is in the calculated area of adjacent voxels A, we assume that a voxel $a \in \phi^{-1}(n)$ exists, which is only nearly out of this area:

$$d_v^{min}(n) = \begin{cases} r * voxelEdgeLength & \text{if } \phi^{-1}(n) \cap A = \emptyset \\ min(d_v(\phi^{-1}(n))) & \text{otherwise} \end{cases}$$

In addition, we compute for each node $n_i \in t$ the effect of removing it from the trajectory t. The resulting trajectory t' is computed and the maximum displacement of the robot arm in workspace $disp_{max}(t, t')$ is determined. This is done iteratively and the node is removed during this process if possible.

Algorithm 1 Shortcut the trajectory to smooth the execution.

function SHORTCUTTRAJECTORY($nodeDistance$ **as** nd, d_{min}, $t = (n_0, n_1, \dots)$)
  $t' \leftarrow t$
  for $n_i \in t$ **do**
    $t'_{tmp} \leftarrow t' \setminus \{n_i\}$
    $d \leftarrow \text{DISP}_{max}(t'_{tmp}, t)$
    $affectedPart \leftarrow \{\text{PREVIN}(t'_{tmp}, n_{i+1}), \dots, n_{i+1}\}$
    ▷ Get all nodes, which are removed from the original trajectory t and the nodes, where the shortcut is between.
    $distances \leftarrow \{nd(n) \colon n \in affectedPart\}$
    $min \leftarrow \text{MIN}(distances)$
    if $min - d < d_{min} \wedge \neg\text{HASSELFCOLLISION}(t'_{tmp})$ **then**
      $t' \leftarrow t'_{tmp}$
    end if
  end for
  return t'
end function

IV. EVALUATION

The evaluation of the DA-DRM algorithm is conducted using the ARMAR-III robot [19] and the ArmarX framework [20]. The results of our DA-DRM algorithm are compared against the DRM and the RRT-Connect [21] algorithm. In the following, we first evaluate the online part of the DA-DRM algorithm. Therefore, the DA-DRM, the DRM and the RRT-Connect algorithm are executed with fixed parameters in different scenes. After this, different roadmaps are compared in terms of calculation time and usability with the DA-DRM algorithm.

To compare the DA-DRM algorithm with the DRM and the RRT-Connect algorithm, two parameters are considered. First, the quality of the generated trajectory is evaluated, which is defined by the smoothness and the feasibility of the trajectory. These are measured in a simulated environment using a model of the ARMAR-III robot. Second, we evaluate the creation time of the trajectories. This is done both in simulation as well as on real-world experiments based on point cloud representation of the scene.

A. Experimental setup

The evaluation is done using the ArmarX framework, which provides a simulation environment in which the scenes, shown in Fig. 5, are created. In the first three scenes the robot stands in front of a table with the arms hanging beside his body. This start configuration as well as the target configuration are randomized in a range of 0.1 radian for all joints, to get a better coverage of different situations. These scenes differ by the objects on the table. The first scene only contains one object to grasp (Fig. 5(a)), which shows a very simple situation. As a more complex scenario, the second scene contains multiple objects (Fig. 5(b)), where the robot has to plan around. The third scene (Fig. 5(c)) uses a plant as barrier.

The fourth scene (Fig. 5(d)) contains three randomly placed cuboids, which are also perceived by the simulated depth sensor. These cuboids are always placed between randomly chosen start and goal configurations. For these configurations it is ensured that they do not violate the safety distance.

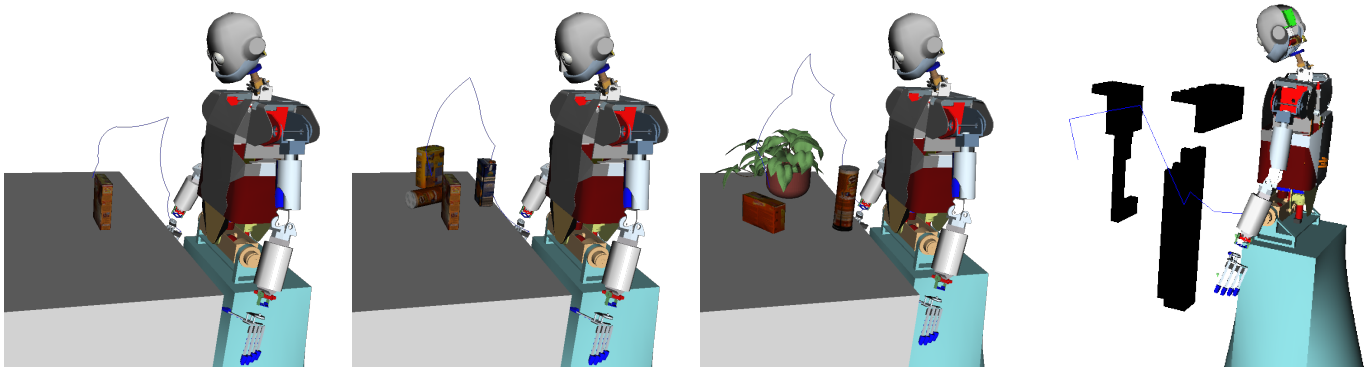
The algorithms are implemented using the ArmarX framework, while the DA-DRM and the DRM algorithms are based on the same source code. Both, the DRM and the DA-DRM algorithm use a roadmap with 16384 nodes and each node is connected to its 20 nearest neighbors. The voxel in both roadmaps have a size of $40\,\text{mm} \times 40\,\text{mm} \times 40\,\text{mm}$. Since both, the DRM and the DA-DRM only use point-cloud data, the RRT-Connect is also executed on this data. Due to the jerky trajectories generated by the RRT-Connect algorithm, a random shortcut post-processor is used additionally for the RRT-Connect evaluation runs. Furthermore, we define an upper bound of calculation time and assume a failure if the runtime for an algorithm exceeds $1000\,\text{ms}$.

For Scene D, see Fig. 5(d) the RRT-Connect and the DRM are also evaluated using a collision checker, which only allows configurations keeping the safety distance. This is done only for the last scene, because for the first three scenes the goal configuration would violate the safety distance.

For the first three scenes, each algorithm is run 250 times and for Scene D 50 different settings are chosen in which each algorithm is run 5 times. For every run the calculation time, the distance in C-space and workspace, and the minimal distance to all surrounding objects are captured. The workspace distance is measured at the Tool Center Point (TCP) of the arm and the C-space distance is calculated as the Euclidean distance of all joint angles in radian along the path:

$$\begin{aligned} d &= \sum_{(a,b) \in Path} euclideanDistance(a, b) \\ &= \sum_{(a,b) \in Path} \sqrt{\sum_{i \in Joints} (a_i - b_i)^2} \end{aligned}$$

In addition to this experiment, where only data from a simulated sensor is used, the DA-DRM algorithm is executed



(a) Scene A (b) Scene B (c) Scene C (d) Scene D

Fig. 5. The scenes used in simulation to compare the algorithms and exemplary TCP trajectories generated with the DA-DRM algorithm are drawn.

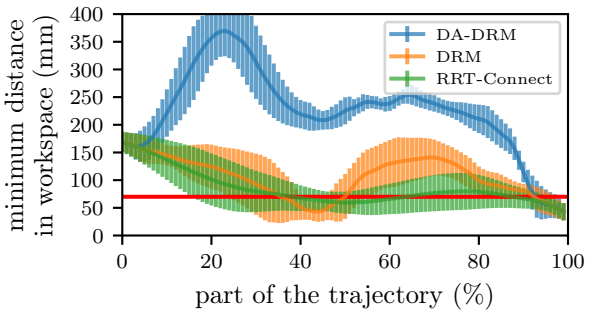
250 times using the real-world point cloud as shown in Fig. 6 with the robot standing in front of the table. The robot is positioned like in the first experiment and the start and goal configurations are randomized by 0.1 radian as well. In contrast to the first experiment, the point cloud is captured on the real robot. Both experiments are run on an Intel(R) Core(TM) i7-7700 CPU running at 3.6GHz which is supported by 16GB of DDR4-2400 RAM.

B. Results

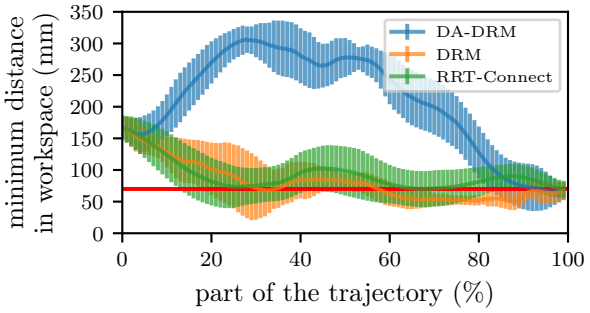
The first experiment, which is run fully in simulation, compares the DA-DRM with the RRT-Connect and the standard DRM. First the quality of these algorithms is compared. A good trajectory connects the start and goal configuration with a short distance for the TCP and minimum joint movements. This first reduces time and energy needed to execute the trajectory and second looks more appealing (i.e. human-like). A second factor for a good trajectory is the feasibility since inaccuracies in the sensor and actor system may require that the trajectory provides a safety distance to obstacles.

For the experiments a safety distance of $d_{min} = 70\,\mathrm{mm}$ is assumed. Fig. 7 depicts the minimal obstacle distances along the paths for scenes A to C. For each percentage of the path, the mean value is printed along with the standard derivation over the 250 runs. In all scenes, the trajectories generated

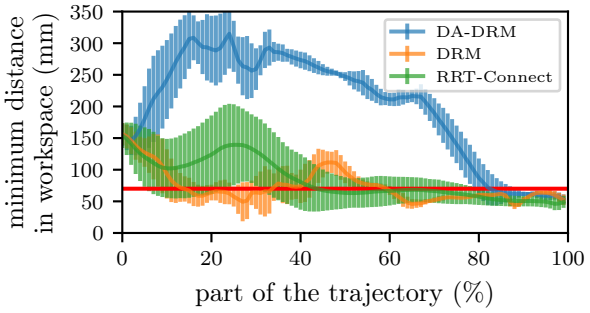
Fig. 6. The point cloud used to evaluate the algorithm on real world data.



(a) Scene A



(b) Scene B



(c) Scene C

Fig. 7. The minimal distance and standard deviation along the trajectory. The red bar visualizes the safety distance, which should be kept.

Algorithm	failed	minimal	average
DA-DRM	1.6 %	3.00 mm	108.20 mm
DRM	0.4 %	0.003 mm	28.81 mm
DRM 70 mm	54.8 %	70.43 mm	107.81 mm
RRT-Connect	5.6 %	0.001 mm	36.25 mm
RRT-Connect 70 mm	81.2 %	70.45 mm	143.70 mm

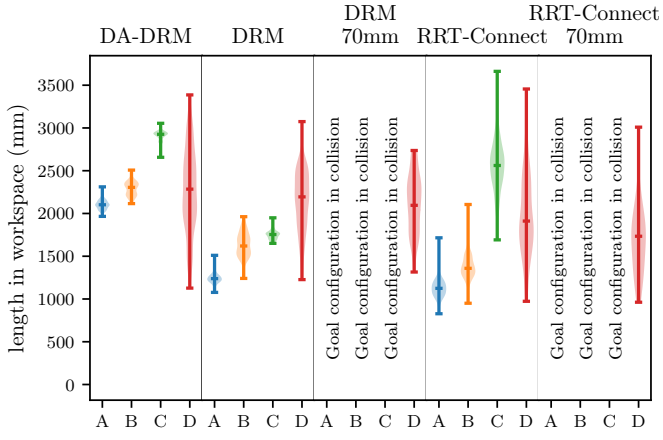
Fig. 8. The minimal distances to the environment evaluated for Scene D (5(d)). For each algorithm the number of failed runs, the minimal and average distance calculated over all runs is shown.

by the RRT-Connect and the DRM fall considerably below the safety distance. This is an expected result, because both algorithms do not include the safety distance in the calculation. In contrast to that, the DA-DRM keeps the safety distance for the whole trajectory and violate this distance in the area close to the goal configuration where the distance should fall below such as distance to achieve the target.

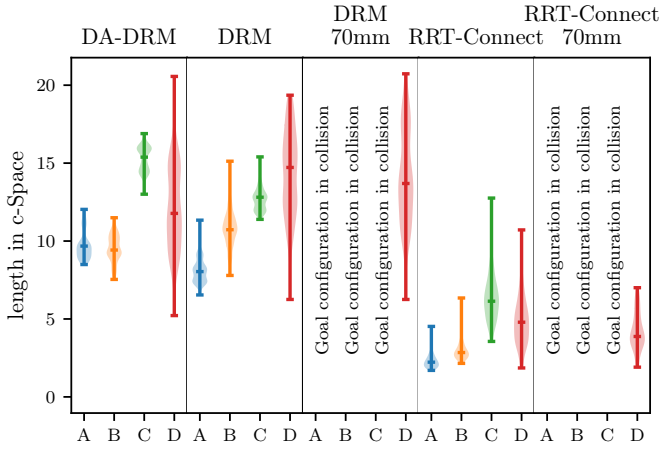
The results for scene D are shown in (Fig. 8). While the RRT-Connect and the DRM again violate the safety distance, their equivalents, which consider the safety distance as blocked area, always keep the safety distance. However, these algorithms have the highest number of failures, since they are not able to plan in narrow passages. The DA-DRM is always able to plan a path in the given time if the adapted roadmap includes a path and follows the safety distance if possible. Thus scene D shows, that the DA-DRM provides a good trade-off between successfully planning a path and ensuring the safety distance.

The smoothness of the trajectories is measured by the distances in workspace and C-space, which are shown in Fig. 9(a) and Fig. 9(b). The data is provided in a violin plot, where on the y-axis for each length the probability density is shown. On the x-axis the results are labeled by the scene number and the algorithm, while the results are sorted by the algorithm. These two diagrams show that the RRT-Connect provides the shortest trajectories in the C-space and distances which are mostly equal to the other algorithms in the workspace. The reason for this is the post-processing algorithm which optimizes for the shortest possible trajectory. For the DA-DRM, Fig. 9(a) shows, that it provides comparatively long trajectories, which is affected by the safety distance. For the C-space, the results are different, as shown in Fig.9(b). The DA-DRM and the DRM algorithm generate trajectories of nearly equal length. This shows, that the DA-DRM algorithm provides smoother trajectories than the DRM algorithm, because trajectories of the DA-DRM algorithm allow more movement in the workspace with similar joint movement. But these trajectories are not as smooth as those generated by the RRT-Connect, which again produces the shortest trajectories. Such short trajectories can not be generated by the DA-DRM since it has to keep the safety distance.

The time needed to generate the trajectories is shown in Fig. 10. The violin plot shows that the DRM takes with under 100 ms the least time. The DA-DRM algorithm needs 200 ms to 500 ms in the mean which is about as much as the RRT-Connect needs. But in terms of consistency the DA-DRM beats the RRT-Connect. The RRT-Connect has much more variance



(a) Trajectory length in workspace.



(b) Trajectory length in C-space.

Fig. 9. The length of the trajectory in the workspace (9(a)) and C-space (9(b)) for the different algorithms in the different scenes.

in the calculation time for each scene and has a high amount of runs which need about 1000 ms in Scene D.

To understand the increase in duration for the DA-DRM against the DRM, we need to have a closer look at the duration of each step during the execution. For this purpose, the time for trajectory generation using a real world point cloud is broken down, in the second experiment. Fig. 11 depicts the duration of each step of the algorithm in a violin plot. As can be seen, the sum of the average duration is about 230 ms, which is the time needed in simulation.

The algorithm is divided into five major steps, while all other calculations need less than 5 ms. First, the *generate distance grid* step needs about 30 ms. During this step, all points of the point cloud are iterated and the corresponding voxels are marked. This includes the iteration over the adjacent voxels, whose number increases cubically with the safety distance. The cubical increase of the considered voxels is the reason for the relative high time consumption. The second step, which needs about 50 ms is the *adapt the graph* step. During this step all marked voxels are iterated and the corresponding nodes and edges are updated. Since the number of marked voxels is high

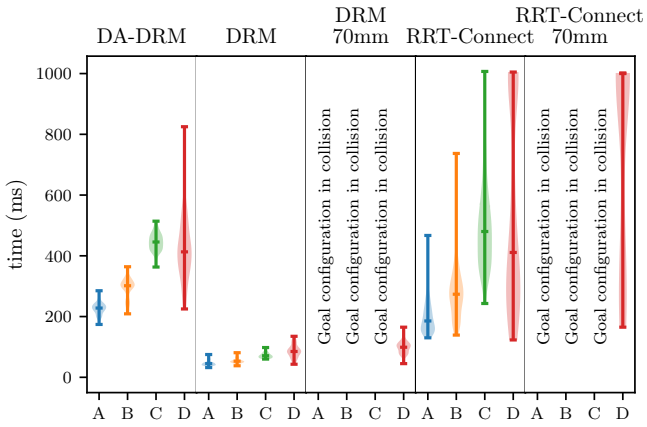


Fig. 10. The calculation times for the different algorithms in the different scenes.

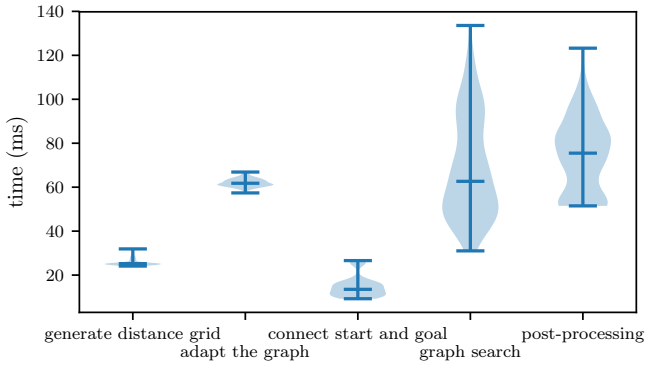


Fig. 11. The calculation time for each step of the DA-DRM using a real world point cloud.

for large safety distances, this step needs a significant amount of time. In the third *connect start and goal* step, which needs about 15 ms, the start and goal configurations are connected to the graph. This includes a k-nearest neighbors search, which is simply done by iterating over all nodes. It could be sped up by using a cover-tree data structure ([18], [22]). The fourth step is the *graph search* step, which only includes one run of the $A^\star$ algorithm. The duration of about 35 to 130 ms is a result of the changes of the edge weights. Because the weights are changed, the optimal path leads not along an edge created by the blocked part of the environment, which increases the search area of the $A^\star$ algorithm. Therefore, the number of nodes considered during a run is highly increased. In the last step, the graph is smoothed during the post-processing, which needs between 40 and 120 ms. To do this, new edges are created, which are not included in the graph. Since the resulting trajectory has to be self collision free, every new edge needs to be checked for self collisions, which is computationally costly.

C. DA-DRM Roadmap Parameter Evaluation

A DA-DRM roadmap has mainly three different parameters to adjust. These are the voxel size, the number of nodes and the number of edges per node. To find the best values, different roadmaps are calculated and then evaluated using the test methods listed in the introduction of this section. As an initial setup for the parameters, the values presented by Kunz et al. [18] are used. They use 16384 nodes, 20 edges per node and a voxel edge length of 50 mm.

For the voxel size, smaller is theoretically better, since this improves the accuracy for the planning. To find a good voxel size, roadmaps with 50 mm, 40 mm and 30 mm voxel edge length, each with 16384 nodes and 20 edges per node are compared. While C-space and workspace length of the generated trajectories do not differ much, the trajectory creation time and the roadmap creation time are getting longer and the roadmap size is getting larger for smaller voxel edge lengths:

Voxel edge length (mm)	**50**	**40**	**30**
Trajectory creation (ms)	100-150	100-230	150-250
Roadmap creation (h:m)	3:38	7:10	18:10
Roadmap size (MB)	859	1386	2637

For a voxel edge length of 50 mm and 40 mm the trajectory creation time is very equal, with both under 200 ms. The time for roadmap creation with 30 mm voxel edge length is much higher, because the number of adjacent voxel increases to cover the safety distance. In addition to the larger trajectory creation time, the roadmap with 30 mm voxel edge length needs more than double the space and time to create the roadmap. The creation time of about 7 hours and the size of 1.3 GB is reasonable for a roadmap. So using a voxel edge length of 40 mm gives an improvement in accuracy with reasonable creation time and space need.

Using a roadmap with more edges tends to generate smoother trajectories, while the roadmap creation time, size and the trajectory generation time are increased. A roadmap with 20 edges per node, as used by Kunz et al. [18], provides good results for the DRM algorithm, while space and time requirements are reasonable, as shown above. For the DA-DRM algorithm, 20 edges per node seems to be a good choice, while less edges provide also reasonable results:

Edges per node	**20**	**15**
Trajectory creation (ms)	100-230	100-240
workspace length	1200-2400	1750-3000
C-space length	8-15	9-17

The number of nodes affects the precision of the trajectory planning in different environments. More nodes increase the probability, that a node is near to the goal configuration. But an increase of nodes also increases the runtime of the $A^\star$, the pre-calculation time and the roadmap size. Since the number of 16384 nodes provides a good precision, this number of nodes is used throughout this work.

V. CONCLUSIONS

In this work, the DA-DRM algorithm is presented, which extends the DRM algorithm by incorporating the distance to obstacles for trajectory generation. The obstacle distance of the planned trajectories has been increased to maintain a safety distance while achieving good performance. The DA-DRM algorithm includes a post-processing step, which uses a created distance grid to smooth the generated trajectories resulting in

shorter trajectories. In our evaluation scenarios, the runtime of the algorithm has been measured with values below $500\,\mathrm{ms}$ which makes it suitable for real-time applications. We also showed that the algorithm is capable to deal with direct visual input (i.e. point clouds) without the need of any computer vision algorithms such as object detection and localization.

To improve the trajectory creation time, the algorithm could be implemented using a field-programmable gate array (FPGA) as this was done for the DRM by Murray et al. [10]. Since all extensions are working with the voxel grid and the W-C mapping, the adaptations to an implementation on the FPGA are easy to conduct.

REFERENCES

[1] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[2] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[3] J. P. Van Den Berg and M. H. Overmars, "Roadmap-based motion planning in dynamic environments," *IEEE Transactions on Robotics*, vol. 21, no. 5, pp. 885–897, 2005.

[4] P. Leven and S. Hutchinson, "Toward real-time path planning in changing environments," in *Workshop on Algorithmic Foundations of Robotics.*, 2000.

[5] ——, "A framework for real-time path planning in changing environments," *The International Journal of Robotics Research*, vol. 21, no. 12, pp. 999–1030, 2002.

[6] M. Kallmann and M. Mataric, "Motion planning using dynamic roadmaps," in *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, vol. 5. IEEE, 2004, pp. 4399–4404.

[7] H. Liu, X. Deng, H. Zha, and D. Ding, "A path planner in changing environments by using wc nodes mapping coupled with lazy edges evaluation," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*. IEEE, 2006, pp. 4078–4083.

[8] L. Jaillet and T. Siméon, "A PRM-based motion planner for dynamically changing environments," in *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 2. IEEE, 2004, pp. 1606–1611.

[9] H. Akbaripour and E. Masehian, "Semi-lazy probabilistic roadmap: a parameter-tuned, resilient and robust path planning method for manipulator robots," *The International Journal of Advanced Manufacturing Technology*, pp. 1–30, 2016.

[10] S. Murray, W. Floyd-Jones, Y. Qi, D. Sorin, and G. Konidaris, "Robot motion planning on a chip," in *Robotics: Science and Systems*, 2016.

[11] Y. Yang, W. Merkt, V. Ivan, Z. Li, and S. Vijayakumar, "HDRM: A resolution complete dynamic roadmap for real-time motion planning in complex scenes," 2017.

[12] Y. Yang, V. Ivan, Z. Li, M. Fallon, and S. Vijayakumar, "idrm: Humanoid motion planning with realtime end-pose selection in complex environments," in *Humanoid Robots (Humanoids), 2016 IEEE-RAS 16th International Conference on*. IEEE, 2016, pp. 271–278.

[13] A. Voelz and K. Graichen, "Distance metrics for path planning with dynamic roadmaps," in *ISR 2016: 47st International Symposium on Robotics; Proceedings of*. VDE, 2016, pp. 1–7.

[14] R. Geraerts and M. H. Overmars, "Creating high-quality paths for motion planning," *The International Journal of Robotics Research*, vol. 26, no. 8, pp. 845–863, 2007.

[15] R. Diankov, "Automated construction of robotics manipulation programs," Ph.D. dissertation, Robotics Institute, Carnegie Mellon University, 2010.

[16] S. Quinlan and O. Khatib, "Elastic bands: Connecting path planning and control," in *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on*. IEEE, 1993, pp. 802–807.

[17] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "CHOMP: Gradient optimization techniques for efficient motion planning," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, 2009, pp. 489–494.

[18] T. Kunz, U. Reiser, M. Stilman, and A. Verl, "Real-time path planning for a robot arm in changing environments," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE, 2010, pp. 5906–5911.

[19] T. Asfour, K. Regenstein, P. Azad, J. Schröder, N. Vahrenkamp, and R. Dillmann, "ARMAR-III: An Integrated Humanoid Platform for Sensory-Motor Control," in *IEEE/RAS International Conference on Humanoid Robots (Humanoids)*, Genova, Italy, December 2006, pp. 169–175.

[20] N. Vahrenkamp, M. Wächter, M. Kröhnert, K. Welke, and T. Asfour, "The robot software framework armarx," *it-Information Technology*, vol. 57, no. 2, pp. 99–111, 2015.

[21] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, vol. 2, 2000, pp. 995–1001.

[22] A. Beygelzimer, S. Kakade, and J. Langford, "Cover trees for nearest neighbor," in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 97–104.