Octree-based, multiresolution environment mapping for efficient calculation of collision-free paths for industrial robots

Benjamin Kaiser, Veronika Breidohr, and Alexandxer Verl

Institute for Control Engineering of Machine Tools and Manufacturing Units (ISW),
University of Stuttgart, Seidenstraße 36., 70174 Stuttgart, Germany
benjamin.kaiser@isw.uni-stuttgart.de,

Abstract. For the automatic calculation of collision-free robot paths, an environment model is required for collision avoidance. The accuracy requirements for the environment model depend very much on the specific task of the robot. The computation time of the collision calculation increases strongly with higher resolution and dimensions of the environment model. For this reason, the demand for high accuracy and fast calculation of the collision-free path is contradictory. To solve this contradiction, we propose a method based on the multi-resolution property of octree based maps. Using a subdivision of the workspace into different subareas and a set of poses to be connected by path planning, a multi-resolution map is created. The individual resolution levels depend on the accuracy requirements of the subareas which are defined by the process requirements. The presented method is evaluated using the "KaBa" path planning framework. For the evaluation, point clouds scanned with a line laser are tested in two path planning scenarios.

Keywords: path planning, multi-resolution map, octomap, industrial robot, collision avoidance

1 Introduction

In production technology, the trend towards individualization of products is playing an increasingly important role. While the complexity of manufactured products is increasing, product life cycles are becoming shorter and shorter. In the case of highly customizable parts, a variant may only be manufactured once over the entire life cycle of the product. Meeting the requirements down to lot size one poses great challenges for production technology. This applies in particular to the cost-effectiveness of manufacturing. Innovative manufacturing systems with a high degree of automation and flexibility are necessary to enable the manufacturing industry to meet the new challenges of the changing market. Industrial robots offer this necessary flexibility to produce small series efficiently. Constantly decreasing acquisition and operating costs also support their use. Nevertheless, industrial robots have so far been used mainly in the automated production of large series such as in the automotive industry and are

rarely found in small series production. The main reason for this is that the existing difficulties in preparing robot tasks prevent their economical use in small series manufacturing. The programming of industrial robots is a difficult task. Despite the use of the CAD/CAM chain, in practice several iterations of robot programming are necessary. This means that the programming time is often longer than the actual production time. This is especially true for small series. In addition, an individual program must be created for each variant. To enable the economic use of industrial robots in small series manufacturing, downtimes resulting from the preparation of tasks such as programming must be reduced or avoided. This can be achieved by automatic path planning for industrial robots. In this case, compliance with the process requirements and avoidance of collisions of the robot with the environment or with itself must be ensured. A complete model of the environment and the robot is required for the planning. Path planning and collision detection should be performed with high accuracy and as fast as possible. However, these two requirements are contradictory. For an exact path calculation, the space must be explored very precisely. At each explored point in space, the inverse kinematics must first be solved to check whether the point can be reached kinematically. For all reachable points, it has to be checked if there are no collisions of the robot with itself or its environment. The calculation of collisions is computationally intensive and depends mainly on the accuracy of the collision models of the robot and its environment. To resolve the contradiction between fast and accurate path calculation, the use of efficient maps as environment models plays a particularly decisive role. Therefore, this paper presents a novel environment model that allows fast and accurate path planning by using different resolutions within the same map.

2 State of the art

Models for 3D environment mapping represent a key role in the automatic path planning of robots. Especially in the field of mobile robotics this topic is intensively investigated. With recent advances of automated path planning for industrial applications like welding or assembly, environment modeling gets more important in the field of industrial robotics as well. Early work mainly focused on the registration of height maps by distance sensors [5]. In this early works, maps were composed of a fixed, two dimensional grid with a height value at each grid point. Multi-height maps followed up on this approach. These can store more than one height value per field and can therefore also display passages [8]. However, since industrial robots move in three-dimensional space and have 6-DOF for approaching poses, this type of map is not suitable for path planning of industrial robots. First approaches for 3D environmental models were based on a fixed grid whose cells represented voxels [6]. However, due to the fixed grid, each field must be stored and initialized whether it is occupied or not. This makes the memory requirements for larger maps very high and hence the approach unsuitable for many applications. In addition to grid-based approaches, the use of point data from distance sensors such as LIDAR or depth cameras is widespread. Here

the map information is stored as point cloud data [1]. The work of [2] deals with multi-resolution maps based on point clouds. The disadvantage of using point clouds is that they do not provide a volumetric representation of the environment. This makes collision detection more complicated since collisions between points can remain undetected. Furthermore, these maps are very susceptible to noise from the used sensor [7]. Using octrees as a data structure to store maps solves the main problem of grid-based maps [9]. By using the octree, the entire map does not have to be initialized. Nodes are gradually inserted into the map. The use of octrees is very memory efficient. Initially Octree based maps only modeled the occupancy of space [3] [10]. The work of [13] and [7] presents a novel probabilistic octree based map called Octomap. The Octomap 3D occupancy mapping framework models occupied, free and unknown space. The Octomap is used as an environment map in a framework for camera-based path planning for industrial robots (KaBa) [12]. The path planning framework automatically calculates collision-free paths for industrial robots. The calculated paths can be executed directly on a robot controller. The KaBa framework first captures a 3D point cloud of the environment and generates an Octomap. The point cloud is used to localize components. This allows the path planning to be adapted to deviations of the component from the target geometry and deviations of the orientation and position. The KaBa framework has the objective of making the use of industrial robots in small series production cost-efficient.

3 Methodology

3.1 Problem Formulation

For a given path planning request consisting of a discrete sequence of n poses $P_i, i \in [0, n - 1]$, a collision-free path must be determined. The path planning calculates a smooth, collision-free path connecting these poses. For each segment $S_{i-1,i}$ between P_{i-1} and P_i , a feasible set is defined. The path must be contained within this subset of the space. The requested path and maximum tolerance are process dependent. A welding or assembly process may require very small tolerances. Whereas the tolerance for a handling task, where the only goal may be the collision-free path, can be selected accordingly large. The computation time of automatic path planning depends mainly on two factors. First, the step size of the path planning has a strong impact on the calculation time [4]. For each pose explored in space, the inverse kinematics of the manipulator must be solved. If the pose can be reached kinematically, collisions must be discarded [12]. Second, the resolution of the environmental model. This has a direct influence on the number of necessary collision checks at each location. Therefore, a reduction of the map resolution directly leads to shorter planning times. However, it also reduces the accuracy of path planning. This may violate the process requirements regarding precision. In order to overcome the contradiction between accuracy and computation efficiency, the environmental model has to be reduced without having a negative impact on planning accuracy.

Based on the segments $S_{i-1,i}$ and the corresponding tolerances δ_i , the area in which the tool center point moves along the planned path is limited. The process takes place in this area. A reduction of the resolution has a direct effect on the process execution. Therefore, the resolution of the environmental model cannot be reduced in this area. Nevertheless, collisions must be checked with the remaining areas of the map. It must be ensured that the robot does not collide with one of its elements during path execution. By splitting the map into several subareas, the overall map size can be reduced without affecting the required accuracy along the path. This reduction reduces the planning time. The environment model, divided into several subareas, is called Multimap. The Multimap is built on octree based maps. There are three reasons for choosing octrees as a basis. First, maps based on an octree as an internal data structure contain several resolutions inherently. This property can be used directly to generate the Multimap. Secondly, an octree represents voxels. A voxel models a volume other than a point cloud. If the resolution of a point cloud is greatly reduced, the distances between the points may become so large that collisions between them cannot be detected. This is not the case with voxels. Third, operations on the octree can be performed very efficiently [11].

3.2 Resolution areas

Subareas in Multimap can be parameterized and are dependent on the desired path and the corresponding process tolerances. Resolution subareas are defined as cylinders with hemispheres at the ends. The subareas are aligned to the path segments $S_{i-1,i}$. Each subarea has a defined voxel size. The figure 1 shows the division of the map into subareas. The parts of the map that are not within any subarea are assigned the parameterized minimum resolution.

Fig. 1: Subarea division

The maximum and minimum resolution, as well as the number of resolution subareas, can be configured. Since the exact voxel sizes are already defined through the underlying octree it is not possible to create the Multimap with arbitrarily sized voxels. Therefore, the desired highest and lowest resolution is

mapped to the nearest, higher resolution contained in the octree. The resolutions of the areas between the minimum and maximum level are interpolated and mapped to an octree depth as well.

The parameterization of the multimap is done by the user based on the task and the setup of the robot. The minimum voxel size is determined based on the process requirements and tolerances. Also, the physical dimensions of the robot and especially the end effector must be considered. If the voxels are chosen too large, a path may not be possible due to collisions with the environmental model. The size of the end effector must also be taken into account when defining the size of the areas. For the number and size of the areas as well as the maximum voxel size, the calculation efficiency and the resulting movement restriction for the robot must be considered.

3.3 Multimap Generation

Each path planning request contains an existing octree as an environment model. For each request, a multimap based on this octree is generated in a pre-processing step. This step consists of two parts. In the first step, the resolution areas around each segment of the requested path are created accordingly to the parametrization. Cylinders are used to specify the different areas because whether a point is contained by a cylinder can be checked very efficiently. However, any set that provides a method to check if a point is contained or not can be used. The calculated resolution areas specify the maximum depth of the octree for each location. The algorithm 1 shows the procedure for determining which nodes of the octree are included in the multimap.

```

input : A octree and subareas
output: A list of nodes that are included in the Multimap

1 for node  $\leftarrow$  first node to last node do
2   the root (first node) is in the multiMap by default;
3   if node is in the multiMap then
4     if depth of node == depth of one of the resolutionAreas then
5       area := the one of the resolutionAreas at this depth ;
6       if node is inside area then
7         mark node with true; // node and children are added
8       else
9         mark node with false; // node is added
10      end
11    else
12      mark node with true;
13    end
14  end
15 end

```

Algorithm 1: Node selection algorithm

First, the algorithm iterates over the complete tree and checks for each node whether it is part of the Multimap or not. All children of a node that is outside the resolution areas are not added to the Multimap. Only the nodes in the depths that correspond to a resolution area have to be checked. The others are set to the minimum parametrized resolution. Since a lot of nodes are ruled out early by not being contained in any resolution area, the determination is fast, even as the number of nodes expands drastically with every depth. The nodes that are part of the multimap are stored in a list.

```

1 Function addToTree (octoMapNode, multiMapNode) input: one node of
   octoMap and multiMap each
2 if octoMapNode is marked and octoMapNode has children then
3   for  $i \leftarrow 0$  to 7 do
4     if child  $i$  of octoMapNode exists then
5       add child  $i$  to multiMapNode;
6       call addToTree (child  $i$  of octoMapNode, child  $i$  of
         multiMapNode);
7     end
8   end
9 end

```

Algorithm 2: Map building algorithm

In the next step, the Multimap is created by building a new octree. The procedure to build the Multimap based on the node list from the first step is shown in algorithm 2. This algorithm constructs a new octree containing the previously identified nodes.

4 Experiments

Since the Multimap is generated for each path planning request based on the already existing octree this step leads to a calculation overhead in comparison to using the existing map. The idea behind the Multimap is to reduce the overall number of voxels per map and hence speed up the path planning due to the reduced amount of collision queries needed.

This section evaluates the effectiveness of the Multimap. For this purpose, two path planning requests are planned and evaluated using the path planning framework "KaBa" from [12]. The framework for camera-based path planning scans the environment and generates an Octomap from the registered point cloud. The framework includes a multi-level collision detection based on bounding volume hierarchy and approximate convex decomposition of the robot geometries for use with the GJK algorithm. Collisions along the requested path are avoided by using algorithms like RRT or A* without violating the defined path tolerances.

Fig. 2: Pathplanning with octoMap and multiMap

The used Octomap contains a box on a pallet. The map was created from a point cloud scanned with a line laser. The first test case is a path along one edge of the box. In the second test case, the robot moves the tool into the box. For this experiment, file-based Octomaps are used. The path planning is done in Matlab. The Octomap import and the generation of the Multimaps are done in C++. The goal of the Multimaps is to speed up the path planning process. To achieve this, the time-saving in collision detection has to be big enough to compensate for the computation time of the Multimaps. If it has almost the amount of nodes that the Octomap has, it cannot speed up the collision detection, and additionally, take a long time to build. In contrast, the Multimaps is especially useful if the important areas are small.

Fig. 3: Test case 1

Three different parametrizations of Multimaps are tested. Multimaps 1 has three resolution areas, with the radii 70mm, 140mm, and 250mm. The minimum requested voxel size is 40 mm. The resolution resulting from that request is 25.6mm. The maximum allowed voxel size is 300mm. This leads to 204.8mm sized voxels. Multimaps 2 uses the same minimal and maximal voxel sizes, but only one resolution area with radius 140. Multimaps 3 uses two resolution areas with radii 100 and 250, but with the requested maximum voxel size of only 150, resulting in the voxel size 102.4mm. The resulting Multimaps of test case

1 are shown in figure 3. The Voxels are colored according to their size. Three resolution areas result in four different voxel sizes. The three voxel sizes in the corresponding areas, and the voxel size outside of any subarea. The resulting maps for test case 2 are shown in figure 4.

Fig. 4: Test case 2

4.1 Reduction of voxels

The reduction of the voxels by the multimap depends on the parameters. This reduction increases with smaller resolution areas. Table 1 shows the number of voxels in each layer of the Octomap and the generated Multimaps. Which layers are compressed depends on the parameterization of the Multimaps. Multimap reduces the number of voxels in the required resolution (depth 7) between 83% and 97,5% compared to Octomap. In the Octomap all resolutions are included and take up memory space, although they are never used. In the Multimap, the resolutions that are finer than necessary are cut.

	1	2	3	4	5	6	7	8	9	10
Octomap	4	8	20	55	196	701	2901	11442	42465	76848
Multimap 1-1	4	8	20	55	57	68	115	0	0	0
Multimap 2-1	4	8	20	55	24	90	425	0	0	0
Multimap 3-1	4	8	20	55	196	132	147	0	0	0
Multimap 1-2	4	8	20	55	99	80	72	0	0	0
Multimap 2-2	4	8	20	55	29	100	479	0	0	0
Multimap 3-2	4	8	20	55	196	218	106	0	0	0

Table 1: Octree nodes per layer in Octomap and Multimap

4.2 Planning times

Table 2 shows the resulting planning times for the two test cases. All times are the averages of six runs. The first column of 2 shows the time to import

the Octomap. This includes reading the binary file, creating the multimap and converting it into a Matlab-compatible format. The import of the Octomap takes the most time, therefore the time difference between the import of the Octomap and the Multimap is only between 0.05481s and 0.0954s. The times for the path planning are different. Multimap times are usually shorter. In test case 1 Octomap is partially faster. The reason for this is that there are no collisions. Altogether the collision calculation takes less time. Since no collisions have to be avoided here, space is only explored along the requested path. Nevertheless, the total planning time, which also includes times for path preprocessing and other additional times, is always shorter with Multimap. Multimap shows an advantage in memory consumption and since its tree is pruned, it can be traversed faster. The use of Multimap accelerates the overall process.

	preprocessing	path planning	total time
Octomap -1	0,23059s	1,2367s	3,1995s
Multimap 1-1	0,29217s	1,40057s	2,38873s
Multimap 2-1	0,30200s	1,19960s	2,18203s
Multimap 3-1	0,28667s	1,72902s	2,90948s
Octomap -2	0,22969s	1,92518s	6,16085s
Multimap 1-2	0,29150s	1,44633s	3,86993s
Multimap 2-2	0,32517s	1,44283s	3,90612s
Multimap 3-2	0,2845s	1,25317s	3,56523s

Table 2: pathplanning times of Octomap and Multimap in comparison

5 Conclusion

This paper presents a method for generating 3D maps for the automatic path planning of industrial robots. The Multimap uses octrees as a basis and allows the direct use of multiple resolution levels of the octree for path planning. The division of the map into subareas of different resolution offers the possibility to solve the contradiction between exact and fast path planning. The first experiments show that the Multimap leads to overall faster planning times despite the calculation overhead. In further steps, the Multimap needs to be examined further using the example of a more complex path planning scenarios with many collisions and space exploration where the benefit of Multimap can be seen more clearly.

Due to the promising first results, the multimap should be integrated into the native implementation of the KaBa path planning framework for subsequent work. The change from the Matlab implementation to the native variant of KaBa allows much shorter planning times and allows experimental validation on a real robot cell. Besides, the question of how the parameterization of the multimap can be carried out in practice by the user must be further investigated.

References

1. Cole, D.M., Newman, P.M.: Using laser range data for 3d slam in outdoor environments. In: Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006, pp. 1556–1563. IEEE (May 15-19, 2006). DOI 10.1109/ROBOT.2006.1641929
2. Dryanovski, I., Morris, W., Xiao, J.: Multi-volume occupancy grids: An efficient probabilistic 3d mapping model for micro aerial vehicles. In: 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1553–1559. IEEE (18.10.2010 - 22.10.2010). DOI 10.1109/IROS.2010.5652494
3. Fournier, J., Ricard, B., Laurendeau, D.: Mapping and exploration of complex environments using persistent 3d model. In: Fourth Canadian Conference on Computer and Robot Vision (CRV '07), pp. 403–410. IEEE (28.05.2007 - 30.05.2007). DOI 10.1109/CRV.2007.45
4. Friedrich, C., Csiszar, A., Lechler, A., Verl, A.: Fast robot task and path planning based on cad and vision data. In: 2017 IEEE International Conference on Advanced Intelligent Mechatronics (AIM), pp. 1633–1638. IEEE (03.07.2017 - 07.07.2017). DOI 10.1109/AIM.2017.8014252
5. Grisetti, G., Stachniss, C., Burgard, W.: Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Transactions on Robotics* **23**(1), 34–46 (2007). DOI 10.1109/TRO.2006.889486
6. Hans Moravec, A. E. Elfes (eds.): *High Resolution Maps from Wide Angle Sonar* (1985)
7. Hornung, A., Wurm, K.M., Bennewitz, M., Stachniss, C., Burgard, W.: Octomap: an efficient probabilistic 3d mapping framework based on octrees. *Autonomous Robots* **34**(3), 189–206 (2013). DOI 10.1007/s10514-012-9321-0
8. Kümmerle, R., Triebel, R., Pfaff, P., Burgard, W.: Monte carlo localization in outdoor terrains using multilevel surface maps. *Journal of Field Robotics* **25**(6-7), 346–359 (2008). DOI 10.1002/rob.20245
9. Meagher, D.: Geometric modeling using octree encoding. *Computer Graphics and Image Processing* **19**(2), 129–147 (1982). DOI 10.1016/0146-664X(82)90104-6
10. Pathak, K., Birk, A., Poppinga, J., Schwertfeger, S.: 3d forward sensor modeling and application to occupancy grid based sensor fusion. In: 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2059–2064. IEEE (29.10.2007 - 02.11.2007). DOI 10.1109/IROS.2007.4399406
11. Srihari Narasimhan, Ralf-Peter Mundani, Hans-Joachim Bungartz: An octree- and a graph-based approach to support location aware navigation services & computing, psc 2006, las vegas, nevada, usa , june 26-29, 2006. pp. 24–30
12. Weseler, P., Kaiser, B., Vrugt, J.t., Lechler, A., Verl, A.: Camera based path planning for low quantity - high variant manufacturing with industrial robots. In: Camera based path planning for low quantity - high variant manufacturing with industrial robots, pp. 1–6. IEEE (20.11.2018 - 22.11.2018). DOI 10.1109/M2VIP.2018.8600833
13. Wurm, K.M., Hennes, D., Holz, D., Rusu, R.B., Stachniss, C., Konolige, K., Burgard, W.: Hierarchies of octrees for efficient 3d mapping. In: 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 4249–4255. IEEE (25.09.2011 - 30.09.2011). DOI 10.1109/IROS.2011.6094571

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.