Design and Evaluation of a Multi-Agent Software Architecture for Risk-Minimized Path Planning in Human-Robot Workcells

Tobias Werner, Dominik Riedelbauch, and Dominik Henrich

Chair for Robotics and Embedded Systems,
Universität Bayreuth, D-95440 Bayreuth, Germany,
tobias.werner@uni-bayreuth.de,
http://robotics.uni-bayreuth.de

Abstract. Close quarters human-robot collaboration promises the integration of human flexibility with robot precision, endurance, and strength. However, such collaboration requires a monitoring solution for the shared human-robot workspace to drive the robot manipulator. Recent research has already developed a variety of individual software components for monitoring solutions, from object detection over reconstruction to path planning. In our contribution, we implement existing components for multi-camera reconstruction, risk estimation and path planning in the form of concurrent agents and we connect these agents through a shared blackboard in order to realize risk-minimal path planning for a robot manipulator. We discuss our software architecture, and we evaluate the final software in an example application of human-robot collaboration. In conclusion, our contribution enables risk-minimal path planning on an industrial manipulator at a 10 Hz update rate.

Keywords: risk-minimized path planning, blackboard path planner, software architecture, shared human-robot workcells

1 Introduction

For past decades, robot manipulators have proven their worth in industrial automation due to their strength, endurance, and precision. Humans, in contrast, excel in cognition and can quickly adapt to unexpected or unplanned situations. Therefore, it seems worthwhile to combine the individual advantages of humans and robots in a shared workspace. Prospects include increased process flexibility and reduced maintenance downtimes.

Close quarters collaboration between humans and robots is only acceptable if robots can perceive and evade human coworkers in the shared workspace. Our particular goal is to find a real-time, risk-minimized path for a robot manipulator in a shared workspace under the presence of moving humans or other a priori unknown obstacles. See Figure 1 for an example application.

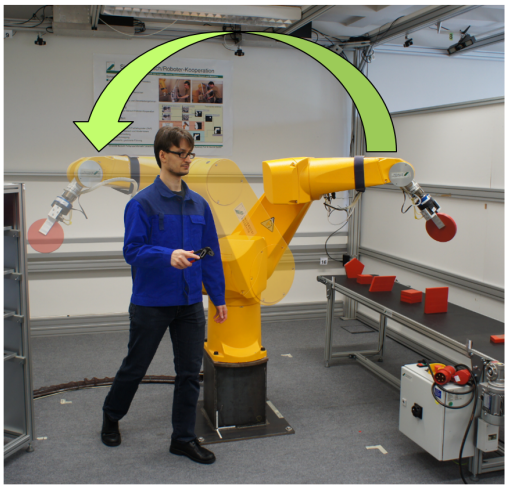

Fig. 1. Our example application: A Stäubli RX130 robot manipulator is to transfer items from the right-hand conveyor belt to the left-hand tool shelf. To minimize the risk of a collision with humans or a priori unknown obstacles in the shared human-robot workspace, the robot perpetually has to adapt its trajectory in real-time.

More formally, we must find a new path $p_t : [0, 1] \to \mathbb{R}^n$ in the n-dimensional configuration space from the current robot position $p_t(0) = q_{t,s}$ to a set goal $p_t(1) = q_e$ at regular update steps $t \in \mathbb{R}$. We express risk for the current update step through a risk function, $r_t : \mathbb{R}^n \to \mathbb{R}$. This function considers detrimental factors, including the probability of object presence within the robot geometry for given joint coordinates. In theory, we seek a path that globally minimizes the risk, $p_t = \text{argmin}_p(\int r_t(p(\lambda))\,\mathrm{d}\lambda)$. In practice, it is impossible to find a globally risk-minimal path. We instead strive for a reasonably risk-minimized path.

In the following, we contribute an alternative software architecture that is tailored specifically to the demands of risk-minimized path planning. Our software architecture is both light-weight and efficient: On the one hand, we rely on concurrent (i.e. distributed or threaded) computations where possible. On the other hand, we combine a multi-agent paradigm with communication through a blackboard to minimize coupling and to maximize cohesion of individual software components. This in turn allows for a light-weight interfacing of software components even over thread or network boundaries. Finally, we choose existing software components that are well-suited for risk-minimized path planning.

The remainder of our contribution is structured as such: Section 2 presents related work, both in terms of software architectures and in terms of software components. Section 3 gives a high-level overview over our software architecture for risk-minimized path planning. Section 4 highlights specific software components within our software architecture. Section 5 evaluates our architecture in an example use case from human-robot collaboration. Section 6 reviews our contribution and concludes with an outlook on future work.

2 Related Work

In general, research proposes manifold reactional and behavioral software components for workspace monitoring in the presence of a priori unknown objects (e.g. speed control and path planning [9]). Each software component requires a specific type of environment representation (e.g. point clouds [31], voxel spaces [25]). The environment representations in turn originate from individual sensors (e.g. artificial skins [26], depth cameras [17], color cameras [11]), or from fusion of data over multiple sensors (e.g. multi-camera systems [19] that find visual hulls [15] or photo hulls [4]). Throughout all software components, there are two distinct variants: approaches either follow a Boolean (e.g. [13]) or a probabilistic (e.g. [23]) paradigm. Boolean variants use discrete input (e.g. object silhouettes) to derive Boolean environments (e.g. object volumes) for collision-free path planning. Probabilistic variants use continuous input (e.g. probabilities for object presence) to derive probabilistic output (e.g. 3D certainty grids) for optimized path planning. Notably, probabilistic variants can incorporate sensor errors, system latencies, or trajectory risks (e.g. [12], [3]).

Each of the above software components runs an intricate and potentially expensive algorithm. To control a robot manipulator, a real-world monitoring solution has to interface a variety of such components while satisfying soft real-time constraints. Existing solutions for this interfacing split into three distinct categories: robot system architectures, knowledge data bases, and world models. Robot system architectures (e.g. ROS [21]) focus on flexible data exchange over software components, but usually prefer stability and extensibility over low-level performance. Knowledge data bases (e.g. RoboBrain [24]) expose extensive semantic knowledge through relational queries, yet do not care much about real-time system latencies. Finally, world models (e.g. OctoMap [31]) enable efficient access to low-level geometric data, but do not explicitly consider integration with threaded or distributed software components.

At implementation level, software architectures tend to follow established software design paradigms (see [8]). As risk-minimized path planning involves software components with expensive and intricate algorithms, design paradigms for threaded or distributed workload are of particular relevance. To achieve implicitly safe, efficient, and extensible parallelization, research proposes a paradigm of minimized coupling and maximized cohesion (see [14]). Multi-agent systems (see [27]), often combined with blackboard communication schemes (see [10]), are a common implementation of this paradigm. In robotics, distributed blackboards predominantly synchronize shared knowledge over mobile robot agents (e.g. [6]) and manage light-weight software components within a single mobile robot (e.g. [28]). Opposed to approaches for mobile robots, ENACT (see [30]) is a software architecture designed specifically for robot manipulators. ENACT successfully combines multi-agent software components with a blackboard in form of a world context to solve a variety of sub-symbolic and symbolic tasks, including collision free path planning or physically grounded pick-and-place operations. Therefore, applying the same paradigms to the computationally expensive task of risk-minimized path planning seems promising.

3 Software Architecture

Our contribution is a novel software architecture specifically tailored to risk-minimized path planning. As such, our contribution stands in contrast to above, more generic architectures. The remainder of this section presents a summary of our overall software architecture, while the next section elaborates on individual software components within our architecture.

At the top-most layer, our software architecture represents a generic pipeline for path planning: A source stage acquires sensor data through a multi-camera system (see Figure 2), a subsequent processing stage derives path segments from sensor data, and a terminal sink stage dispatches path segments to the robot manipulator for execution. Our software architecture implements all three stages by means of a distributed processing model. The source stage distributes over a series of *Camera PCs*, each occupied with expensive preprocessing of images from a single camera. The processing stage concurrently performs risk-minimized path planning over preprocessed sensor input on a single *Server PC*. Finally, an independent *Controller PC* drives a robot manipulator with resulting path segments. This distribution model follows the paradigm of minimized coupling and maximized cohesion. Calculations on individual PCs are mostly independent and thus do not couple, while the tightly coherent state of path planning remains local to the Server PC. Transfers in-between PCs in turn require but light-weight and efficient interfaces. In the context of our overall software architecture, distribution yields two major advantages: On the one hand, distribution enables us to offload expensive preprocessing to separate PCs, a vital contribution to real-time update rates. On the other hand, distribution hardens our architecture against failure of individual software or hardware components.

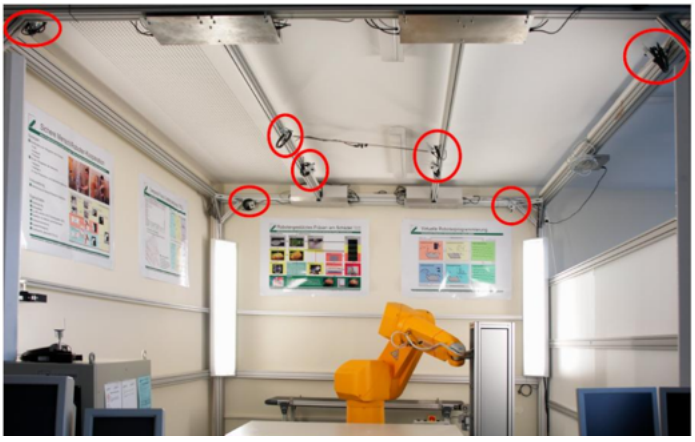

Fig. 2. To detect humans or obstacles within the shared human-robot workspace, we use a multi-camera system. This system consists of multiple intrinsically and extrinsically calibrated color cameras (red circles).

At a layer below the distributed model, our software architecture employs threaded software components to take advantage of the multi-core CPU in the Server PC. A standalone *3D Reconstruction* component performs sensor fusion on any camera images that Camera PCs transmit over network. The other software components on the Server PC are concerned with path planning and path execution. These software components collaborate under a multi-agent paradigm: A blackboard holds path suggestions, which both path planning and path execution agents read and write to. Path planning agents evaluate current results of sensor fusion to derive new path suggestions, to modify existing suggestions, or to discard stale suggestions. While the choice of planning agents — and of respective software components — is flexible, there are three mandatory agents: Threaded *Path Annotation* synchronizes risk estimates of stale blackboard suggestions to current fusion results. *Path Selection* uses annotated risks to perpetually choose a risk-minimized path from robot position $q_{t,s}$ to joint-space goal q_e. Finally, *Path Execution* pulls pending segments of the currently selected path from the blackboard and transmits these to the *Controller PC* for execution by the robot manipulator. Figure 3 illustrates software components in our software architecture and shows data flow between individual components.

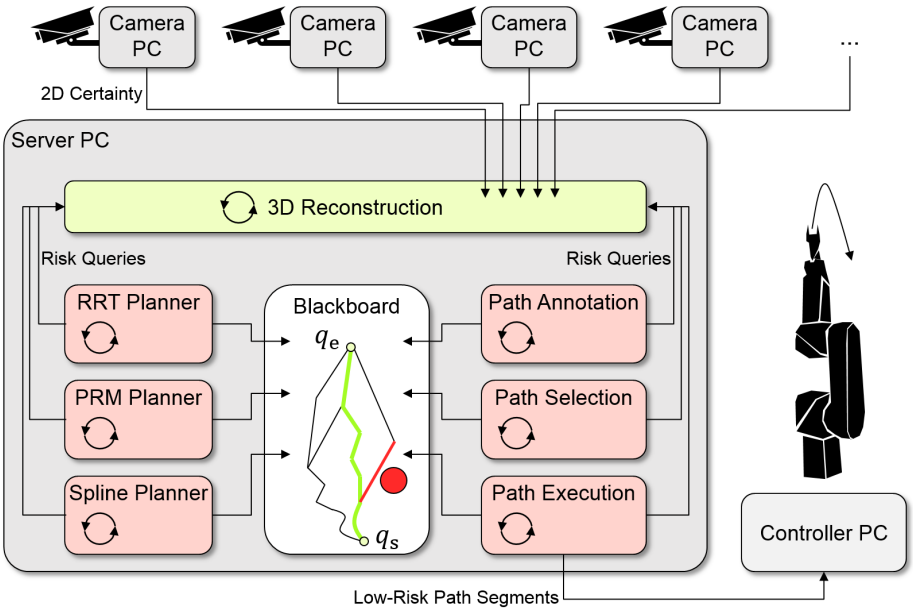

Fig. 3. Overview over software components and data flow in our software architecture. *Camera PC*s, the *Server PC*, and the robot *Controller PC* follow a distributed model, while arrow cycles indicate multi-threaded software components on the Server PC. The shared blackboard currently holds three path suggestions. The red path segment implies high risk due to proximity of an exemplary object. The green path yields minimized risk and thus has been chosen for execution.

4 Software Components

In this section, we provide details on some of the software components that have been introduced in the preceding section. Due to limited scope, we refer to original sources where appropriate. As we desire to find a risk-minimized robot path, we are restricted to software components that can handle probabilistic data. In particular, probabilistic data permeates our entire pipeline from image preprocessing on the Camera PCs to path selection on the Server PC. We perform a deterministic choice but at the last possible step, directly before dispatching a single path segment to the Controller PC for subsequent execution.

The first software component in our probabilistic pipeline performs image preprocessing on distributed Camera PCs and transforms RGB images from the multi-camera system into 2D certainty maps. The respective algorithm consists of two sequential stages: In the first stage of preprocessing, a neural network estimates foreground object probabilities through background subtraction as proposed in [7]. Over time, the neural network collects per-pixel foreground-background information and models pixel statistics within its edge weights. The neural network then compares incoming input images with the current background model to derive independent per-pixel foreground probabilities. To avoid forgetting static objects (i.e. to suppress the sleeping person problem), we extend the original implementation by a modified conditional learning method, where a variable learning rate depends on the background probability of the final pixel. In the second stage of preprocessing, we exploit a priori knowledge to enhance results of the first stage. We conclude that per-pixel probabilities must exhibit spatial and temporal homogeneity except in the vicinity of image edges. We thus consider spatial and temporal inter-pixel coherence to bias object detection. To this end, we apply a Sobel filter to find edges on RGB images. We then pass edges from the current and the previous frame into another neuronal network to approximate an energy-minimal distribution over probabilities from the first preprocessing stage. See Figure 4 for example results.

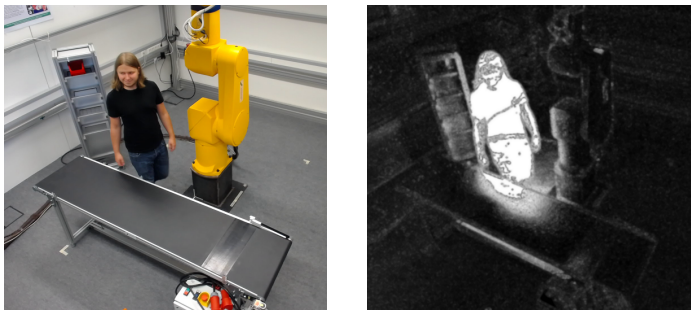

Fig. 4. Left: High-resolution image from the multi-camera system. Right: 2D certainty map with foreground probabilities. Note that these images do not show additional a priori knowledge, including the current manipulator volume or static workspace objects.

The Camera PCs transfer final 2D certainty maps to the Server PC over TCP/IP network. Quantification and run-length encoding avoid network congestion, while an explicit pull paradigm prohibits unnecessary transfers.

Concurrently to pending network transfers, the software component for 3D reconstruction performs sensor fusion on a synchronized set of 2D certainty maps. The goal of sensor fusion is to build a 3D certainty map with probabilities of object presence for all positions within the workspace. Our implementation closely follows the hierarchical and incremental paradigms presented in [29]: Build quadtrees over input maps, and generate an octree over the workspace through efficient quadtree lookups. During this process, track changes in input maps to maintain unchanged octree branches for efficiency reasons. The original fusion approach [29] only considers binary object presence, so we must adapt this strategy to integrate probabilistic data. To this end, we maintain upper and lower bounds on leaf probabilities within inner quadtree nodes. For each node of the octree in the workspace, we consider camera projections into quadtrees to determine upper and lower bounds of 2D probabilities. At this point, we assume independent probabilities over 2D certainty maps, which allows us to find 3D probabilities by multiplying all 2D probabilities. Finally, we split octree nodes once the difference between lower and upper probability bounds exceeds a set threshold. The resulting octree encodes 3D object probabilities over the workspace. Subsequent software components do not access this octree directly. The 3D reconstruction component instead exposes a risk query that samples the octree over the robot volume which corresponds to a given configuration-space position. This risk query acts as an integral part of the risk function r_t.

The final software components concurrently handle path planning and path execution while collaborating over the shared blackboard. We initialize our blackboard with a single, directed path segment from the robot position $q_{0,s}$ to the goal position q_g. Path planners then iteratively add, execute, or delete segments until the robot has reached its goal position. We have selected and implemented three run-of-the-mill path planners for use in our software architecture: The *RRT Planner* uses rapidly-exploring random trees and adheres to the original RRT proposal [16]. The *PRM Planner* searches for paths on a probabilistic roadmap and closely follows the implementation in [9]. The *Spline Planner* generates paths by applying bezier splines of varying parameters. All three path planners have been adapted to collaborate through the blackboard mechanism. Notably, planners concurrently pick segments from the blackboard and apply their respective planning strategy to determine a more risk-minimal alternative — as estimated through per-node risk queries on 3D reconstruction. Virtues of individual planners govern segment picking. For instance, the RRT planner is prone to picking long segments, while the spline planner concentrates on sharply bent segment pairs. In the end, collaboration of path planners results in a series of path suggestions on the blackboard. *Path Selection* implements a common Dijkstra variant for directed acyclic graphs to perpetually choose a risk-minimal path from these suggestions (e.g. as in [5]). The remaining, mandatory software components realize intuitive functionality and are not discussed here.

5 Experiments

We have evaluated our software architecture in the form of a prototype implementation. In our prototype, a multi-camera system monitors the workspace around a Stäubli RX130 robot with eight inexpensive, consumer-grade Logitech C930e cameras. We distribute preprocessing of incoming 1920x1080 images to 2D certainty images over one PC per camera, where low-end NVIDIA GTX750 GPUs carry most of the workload. A Server PC with an Intel Core i5-4670 CPU polls preprocessing results over Gigabit Ethernet and performs both reconstruction and path planning. 3D reconstruction runs in parallel over two threads, while another six threads correspond to path planning agents. With this setup, we are able to feed risk-minimized path segments into the robot at a 10 Hz rate.

In the context of system performance, we deliberately use a 10 Hz rate to delimit soft real-time capabilities. Three factors contribute to this choice: First, we estimate that overall system latency (e.g. due to USB stacks on the camera PCs, network latency, or processing in the — albeit dated — RX130 CS7 robot controller) has the same order of magnitude. An improvement over the indicated 10 Hz software rate will thus not peculiarly affect overall performance. Second, a 10 Hz update rate is well below the human reactions rate of at most 5 Hz, which enables seemingly instant reactions to human actions within the workcell. Third and last, typical human actions within our workcell imply a limb displacement of at most 2 m/s (see [22] for walking and [18] for pointing gestures). Artificially biasing risk within a respective 20 cm boundary around the current robot volume takes such movements into account without impairing large-scale path planning. Still, one must note that our software architecture is a monitoring solution for risk-minimized path planning and not a safety solution. Additional measures (e.g. an artificial skin [2], a soft manipulator [1], or explicit human tracking [20]) are required to ensure the safety of humans within the shared workcell.

6 Conclusion

In the preceding, we have contributed a software architecture that enables risk-minimized path planning on a robot manipulator. Path planning executes at a soft real-time rate of 10 Hz. Our software architecture achieves this through distributed preprocessing of 2D certainty maps, and through a blackboard paradigm for multi-agent reconstruction, path planning, and robot control.

In future work, we intend to integrate additional safety measures — including an artificial robot skin — into risk minimization. We also plan to integrate risk-minimal path planning into a more involved scenario of flexible human-robot collaboration: In this scenario, task planning can incorporate path risks, which enables the robot choose a risk-minimal goal from a selection of pending tasks.

References

1. A. Albu-Schaffer et al.: *Soft robotics*, Robotics and Automation Magazine, vol. 15, no. 3, 2008.
2. F. Bergner, E. Dean-Leon, G. Cheng: *Event-based signaling for large-scale artificial robotic skin - realization and performance evaluation*, Intelligent Robots and Systems, 2016.
3. L. Blackmore et al.: *A probabilistic approach to optimal robust path planning with obstacles*, American Control Conference, 2006.
4. A. Broadhurst, R. Cipolla: *A Statistical Consistency Check for the Space Carving Algorithm*, British Machine Vision Conference, 2000.
5. T. Cormen et al.: *Introduction to Algorithms, Section 24.3: Dijkstra's algorithm*, ISBN 0-262-03293-7, MIT Press, Second Edition, pp. 595-601, 2001.
6. G. Brzykcy et al.: *Multi-agent blackboard architecture for a mobile robot*, Intelligent Robots and Systems, 2001.
7. D. Culibrk et al.: *Neural Network Approach to Background Modeling for Video Object Segmentation*, Transactions on Neural Networks, 2011.
8. E. Gamma et. al.: *Design Patterns. Elements of Reusable Object-Oriented Software.*, ISBN 978-0201633610, Prentice Hall, 1994.
9. T. Gecks: *Sensorbasierte, echtzeitfähige Online-Bahnplanung für die Mensch-Roboter-Koexistenz*, PhD thesis, Universität Bayreuth, 2011.
10. B. Hayes-Roth: *A blackboard architecture for control*, Artificial Intelligence, vol. 26, no. 3, pp. 251-321, July 1985.
11. S. Kuhn: *Multi-view reconstruction in-between known environments*, Technical report, Universität Bayreuth, 2010.
12. B. Lacevic, P. Rocco: *Towards a complete safe path planning for robotic manipulators*, Intelligent Robots and Systems, 2010.
13. A. Ladikos, S. Benhimane, N. Navab: *Efficient Visual Hull Computation for Real-Time 3D Reconstruction using CUDA*, Computer Vision and Pattern Recognition Workshops, 2008.
14. M. E. Latoschik and H. Tramberend: *A scala-based actor-entity architecture for intelligent interactive simulations*, Software Engineering and Architectures for Realtime Interactive Systems, 2012.
15. A. Laurentini: *The Visual Hull Concept for Silhouette-Based Image Understanding*, Transactions on Pattern Analysis and Machine Intelligence, 1994.
16. S. M. Lavalle: *Rapidly-exploring random trees: A new tool for path planning*, Computer Science Department, Iowa State University, Technical Report, p. 98-11, 1998.
17. C. Lenz et al.: *Fusing multiple kinects to survey shared human-robot workspaces*, Technical Report TUM-I1214, Technische Universität München, 2012.
18. R. Marteniuk et al.: *Constraints on human arm movement trajectories*, Canadian Journal of Psychology, vol. 51, no. 3, pp. 365-378, 1987.
19. A. Ober-Gecks, M. Hänel, D. Henrich, T. Werner: *Fast multi-camera reconstruction and surveillance with human tracking and optimized camera*

configurations, International Symposium on Robotics, 2014.

20. A. Ober-Gecks, D. Henrich, M. Zwicker: *Efficient graphics processing unit-based voxel carving for surveillance*, Journal of Electronic Imaging, vol. 25, no. 4, 2016.
21. M. Quigley et al.: *ROS: an open-source Robot Operating System*, International Conference on Robotics and Automation, 2009.
22. H.J. Ralston et al.: *Energy-speed relation and optimal speed during level walking*, Internationale Zeitschrift fuer angewandte Physiologie einschliesslich Arbeitsphysiologie, vol. 17, no. 4, pp. 277-283, 1958.
23. J. Salvador, J. R. Casas: *Shape from Probability Maps with Image-Adapted Voxelization*, Workshop on Multi-camera and Multi-modal Sensor Fusion Algorithms and Application, 2008.
24. A. Saxena et al.: *RoboBrain: Large-Scale Knowledge Engine for Robots*, arXiv:1412.0691, 2014.
25. D. Stengel, T. Wiedemann, and B. Vogel-Heuser: *Efficient 3d voxel reconstruction of human shape within robotic work cells*, Mechatronics and Automation, 2012.
26. J. Ulmen, M. R. Cutkosky: *A robust, low-cost and low-noise artificial skin for human-friendly robots*, International Conference on Robotics and Automation, 2010.
27. G. Weiss: *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, ISBN 978-0262731317, The MIT Press, 2000.
28. J. Wen et al.: *Multi-agent based distributed control system for an intelligent robot*, Services Computing, 2004.
29. T. Werner, D. Henrich: *Efficient and Precise Multi-Camera Reconstruction*, International Conference on Distributed Smart Cameras, 2014.
30. T. Werner et al.: *ENACT: An Efficient and Extensible Entity-Actor Framework for Modular Robotics Software Components*, 47th International Symposium on Robotics, 2016.
31. K. M. Wurm et al.: *OctoMap: A Probabilistic, Flexible, and Compact 3D Map Representation for Robotic Systems*, International Conference on Robotics and Automation, 2010