Waverider: Leveraging Hierarchical, Multi-Resolution Maps for Efficient and Reactive Obstacle Avoidance

Victor Reijgwart*, Michael Pantic*, Roland Siegwart, Lionel Ott

Abstract— Fast and reliable obstacle avoidance is an important task for mobile robots. In this work, we propose an efficient reactive system that provides high-quality obstacle avoidance while running at hundreds of hertz with minimal resource usage. Our approach combines wavemap, a hierarchical volumetric map representation, with a novel hierarchical and parallelizable obstacle avoidance algorithm formulated through Riemannian Motion Policies (RMP). Leveraging multi-resolution obstacle avoidance policies, the proposed navigation system facilitates precise, low-latency (36ms), and extremely efficient obstacle avoidance with a very large perceptive radius (30m). We perform extensive statistical evaluations on indoor and outdoor maps, verifying that the proposed system compares favorably to fixed-resolution RMP variants and CHOMP. Finally, the RMP formulation allows the seamless fusion of obstacle avoidance with additional objectives, such as goal-seeking, to obtain a fully-fledged navigation system that is versatile and robust. We deploy the system on a Micro Aerial Vehicle and show how it navigates through an indoor obstacle course. Our complete implementation, called *waverider*, is made available as open source[1].

I. INTRODUCTION

Reactive, precise, and reliable obstacle avoidance is vital for mobile robots to safely and efficiently navigate through changing or partially unknown environments. Since obstacle avoidance is an always-on process, it must use minimal computational resources and seamlessly integrate with the robot's other tasks. Existing approaches range from simple reactive methods using 1D distance sensors to optimization-based systems requiring complete 3D maps and vary in complexity, reaction time, and obstacle resolution. While collision avoidance systems that operate directly on raw sensor data may exhibit exceptionally low latency, they can only guarantee safety with respect to consistently observed obstacles within the Field of View (e.g. [1]). One way to introduce memory without losing generality is to use volumetric maps. They can model obstacles of arbitrary shape and explicitly distinguish free and unobserved space. Volumetric maps are well suited to ensure safety even in unknown environments. However, fixed-resolution volumetric mapping frameworks tend to suffer from excessive memory overheads and latency. These can be overcome by using hierarchical volumetric representations such as octomap [2], UFOMap [3], supereight [4], or wavemap [5]. While several works investigated the use
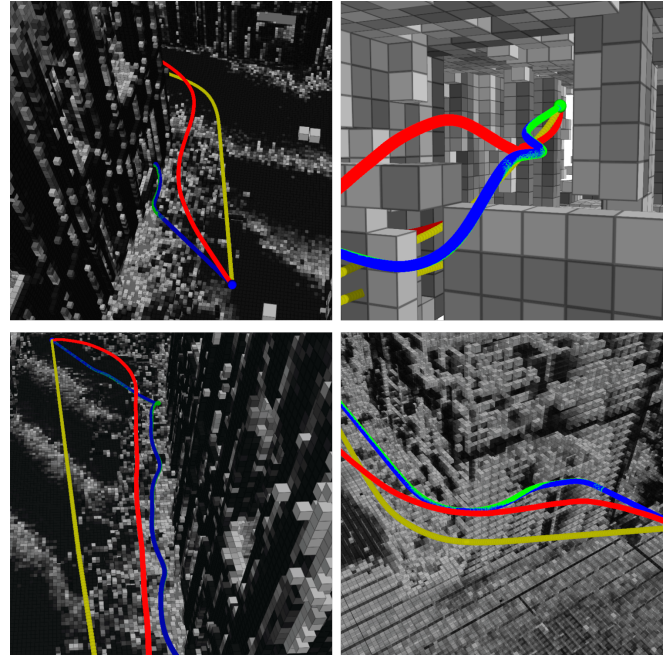
Fig. 1. Example trajectories comparing our multi-resolution collision avoidance method (red) to equivalent RMP-based formulations that consider all obstacles at the highest resolution within a radius of 1 m (green) and 3 m (blue). The fixed-resolution RMP trajectories are jerkier and more prone to get stuck (top-left). CHOMP (brown) yields smooth, albeit overly cautious trajectories and occasionally cuts through obstacles (top-right, bottom-left).

of hierarchical maps for global path planning, most collision avoidance systems still process all obstacles at the highest resolution. Yet, intuitively, one would expect that distant obstacles could be considered at a lower resolution than nearby ones without significantly affecting the robot's behavior. We use Riemannian Motion Policys (RMPs) [6] to formulate a navigation algorithm that is inherently multi-scale and hierarchical. RMPs are purely reactive in nature, and as such, can be formulated extremely efficiently and executed with low latency at controller frequency. Other sampling- or optimization-based methods often need pre- and post-processing steps such as the generation of an Euclidian Signed Distance Field (ESDF) or trajectory smoothing. Conversely, RMPs are formulated as second-order dynamical systems and directly output accelerations, which typically leads to gradual changes and smooth paths. RMPs have some similarities to the well-known potential fields [7], but are a much more expressive framework due to the inclusion of the Riemannian metric that modulates each policy's strength and directionality. In this work, we develop a reactive and safe obstacle avoidance method using RMPs [6] that is tailored to hierarchical volumetric map representations. We numerically

analyze the effects of obstacle resolution on the policy's approximation error as a function of the distance between the robot and the policy. Based on this analysis, we derive a function that computes the ideal resolution for querying the map at a given distance from the robot – allowing us to balance computational effort and accuracy. Using this function, we develop an algorithm that efficiently generates multi-resolution avoidance policies from a hierarchical map. The contributions of this paper are:

- An efficient hierarchical obstacle policy generation algorithm;
- Numerical analysis of the approximation error induced by hierarchical navigation policies;

The correctness of the numerical analysis is statistically validated through a large number of experiments in simulation. Extensive comparisons with baselines and CHOMP [8] demonstrate the favorable run-time and efficiency of our method. Finally, we demonstrate real-world applicability by deploying our system onboard an MAV running at $200\,\text{Hz}$.

II. RELATED WORK

A core decision in any obstacle avoidance system is the environment representation. State-of-the-art systems combine a volumetric map such as a truncated signed distance field [9, 10] or an octree-based occupancy map [2–5] with either a search-based method such as A*[11], a sampling-based approach such as RRT [12], or an optimizer such as CHOMP [8, 13]. All of these methods are comparably slow, as the mapping-planning cycle has multiple performance bottlenecks, and the sampling or optimization steps often rely on post-processed maps. Recently, end-to-end learning-based methods were shown to be effective for collision avoidance [14]. However, their data-driven nature still comes with a lack of generalizability across different environments, sensors, and robot dynamics. Reactive approaches that operate directly on volumetric maps or even raw LiDAR data exist [15], but these methods have considerable memory and computing requirements due to their dense data representation. Although hierarchical volumetric maps have received considerable attention from the planning community, most works focused on global planning [16–18]. Multi-resolution anytime planners [19] have been proposed that bridge the gap to local planning. However, their global context makes achieving the update rates required for low-latency reactive collision avoidance challenging in 3D. Nils et al. [20] propose a full planning pipeline that leverages multi-resolution for efficient orientation-aware planning in environments with very narrow openings. However, their evaluations are performed on pre-computed static maps without perception in the loop, which makes it difficult to judge the system's latency in a reactive collision avoidance setting. Closest to our work is the hierarchical collision avoidance system presented by Goel et al. [21] that adapts the map resolution based on the motion primitives considered by the planner. The method is used in a teleoperation setting and shows promising results in simulated and real environments. However, a significant part of the system's

efficiency results from using a bespoke, purely local map representation whose resolution is set by the planner, which is harder to reuse for additional tasks, including global planning. In comparison, our system achieves comparable efficiency levels using generic hierarchical occupancy maps. This is explained by the efficiency of RMPs, and the fact that our method does not rely on expensive ESDFs. One final benefit of our proposed architecture, compared to both [20, 21], is its high degree of modularity. Formulating obstacle avoidance as a motion policy makes it easy to combine with other policies representing additional objectives such as goal-seeking, visual servoing, or aerial manipulation.

III. METHOD

In the following sections, we describe our approach to efficiently extract multi-resolution obstacle avoidance policies from hierarchical maps and how they integrate with high-level task policies. Figure 2 shows the main parts of the system, consisting of: 1) a volumetric, hierarchical map representation (Section III-A), 2) an algorithm for obstacle extraction (Section III-B), 3) an RMP-based reactive navigation system (Section III-D). For each obstacle cell extracted in 2) an individual obstacle avoidance policy is generated (Section III-C), and combined with all other policies through the RMP framework.

A. Hierarchical map

The proposed method is compatible with any hierarchical occupancy mapping framework, e.g. [2–4]. We chose to use wavemap [5], as it simultaneously achieves state-of-the-art accuracy, memory, and computational efficiency. In a similar fashion to other methods, wavemap leverages octrees to achieve this efficiency. However, instead of storing absolute occupancy values, each node stores Haar wavelet coefficients. Using wavelets achieves significant compression and, more importantly, guarantees that all resolution levels are implicitly synchronized and always up to date. An efficient coarse-to-fine measurement integration algorithm allows wavemap to integrate depth measurements with low latency, even on computationally constrained platforms.

B. Obstacle cell extraction

As will be substantiated in Section IV, reducing the resolution of obstacles as the distance to the robot increases does not introduce significant approximation errors. By representing obstacles at the appropriate resolution, it is therefore possible to efficiently consider fine nearby obstacles and the broader spatial context simultaneously. In this section, we present a hierarchical algorithm that efficiently gathers multi-resolution obstacles by traversing the map in a coarse-to-fine manner. The algorithm (Algorithm 1) starts at the lowest resolution level (root node) of the map and recursively visits each node's higher-resolution children. The algorithm stops expanding a node when that node either has no children or its distance d to the robot exceeds $d_{max}(\lambda)$. We use $d_{max}(\lambda) = 3^{\lambda/3} - 0.25$ where λ corresponds to the node's
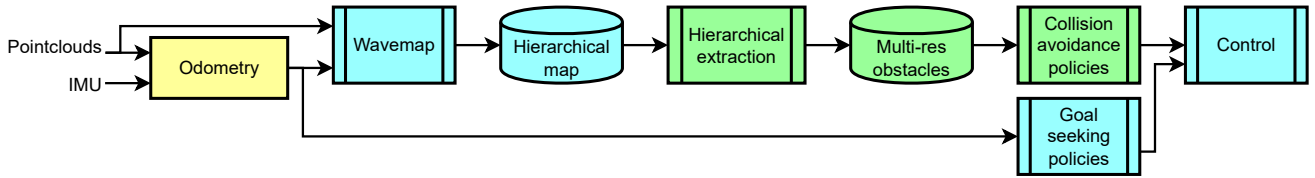
Fig. 2. Block diagram of the proposed navigation system. External components are highlighted in yellow, tightly integrated components in blue, and new components introduced in this paper in green.

Algorithm 1: Hierarchical obstacle extractor

Input: Hierarchical occupancy map $\mathcal{M}$,
  Robot position $\mathbf{p}$
Output: Set of multi-resolution obstacles $\mathcal{O}$

```
1  Function RecursiveExtractor(V, p) is
2  |   d ← ||V_center − p||_2
3  |   if d_max(V_λ) < d then
4  |   |   if IsOcc(V) or HasOccChild(V) then
5  |   |   |   O.insert(V)
6  |   |   end
7  |   |   return
8  |   end
9  |   if not HasOccChild(V) then
10 |   |   return
11 |   end
12 |   for V_child ∈ V do
13 |   |   RecursiveExtractor(V_child, p)
14 |   end
15 end
   // Initialize and start recursion
16 V_root ← GetOctreeRoot(M)
17 O ← RecursiveExtractor(V_root, p)
```
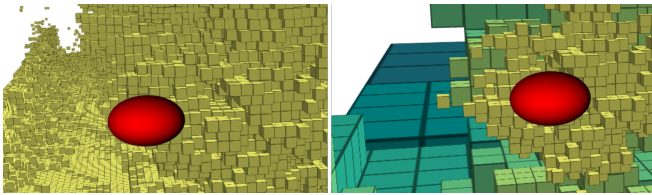

Fig. 3. Comparison of an environment represented using fixed-resolution (left) and hierarchical obstacle cells (right). Our approach uses hierarchical cells, whose resolution (light brown to dark green) is high close to the robot (red) and decreases with distance.

height in the octree[2]. Once such a terminal node is found, an obstacle cell is created if the node or any of its children is occupied. Figure 4 visualizes $d_{max}(\lambda)$ and the resulting maximum distance up to which obstacles are included.

C. Collision avoidance policy generation

For each obstacle cell returned by the previously described algorithm, an individual obstacle-avoidance policy $\mathcal{P}$ [6] is created. In the following, we give a short summary of the most important aspects of motion planning using RMP, however for more details and complete formulas of helper functions we refer to the original text [6]. A policy $\mathcal{P}$ consists of an acceleration function $\ddot{\mathbf{x}} = f(\mathbf{x}, \dot{\mathbf{x}}) \in \mathbb{R}^3$ and a Riemannian metric $\mathbf{A}(\mathbf{x}, \dot{\mathbf{x}}) \in \mathbb{R}^{3 \times 3}$, where $\mathbf{x} \in \mathbb{R}^3$ refers to the robot's current position. The function f drives the robot according to the policy, while the Riemannian metric $\mathbf{A}$ defines a (possibly directional or isotropic) weight of the policy in comparison to other policies. Following

[2]A height of 0 corresponds to the highest resolution/smallest voxel size.

[6], multiple policies $\{\mathcal{P}_0, \ldots, \mathcal{P}_N\}$ can be summed into an equivalent policy $\mathcal{P}_C$ using

$$\mathcal{P}_c = (\mathbf{f}_c, \mathbf{A}_c) = \left(\left(\sum_i \mathbf{A}_i \right)^+ \sum_i \mathbf{A}_i \mathbf{f}_i, \sum_i \mathbf{A}_i \right). \quad (1)$$

We use the obstacle avoidance repulsor from [6] as a policy template for each found obstacle cell. It is formulated as a combination of a pure repulsor $\mathbf{f}_{rep}$, a velocity-dependent damper $\mathbf{f}_{damp}$, and a metric (weight) that becomes 0 if the robot's velocity does not point towards the obstacle. The repulsor is defined as

$$\mathbf{f}_{rep}(\mathbf{x}, \mathbf{r}, d) = \eta_{rep} \exp\left(-\frac{d}{v_{rep}} \right) \mathbf{r}, \quad (2)$$

where d is the distance to the obstacle, $\mathbf{r}$ is the unit vector pointing from the obstacle to the robot, and η_{rep} and v_{rep} are tuning parameters to set the repulsor strength (η_{rep}) and scaling (v_{rep}). Similarly, the damper is defined as

$$\mathbf{f}_{damp}(\dot{\mathbf{x}}, \mathbf{r}, d) = \eta_{damp} \Big/ \left(\frac{d}{v_{damp}} + \epsilon \right) \cdot \mathbf{P}_{obs}(\dot{\mathbf{x}}, \mathbf{r}), \quad (3)$$

again with η_{damp} as a strength parameter and v_{damp} as a scaling parameter. ϵ is a sufficiently small constant to ensure numerical stability. $\mathbf{P}_{obs}(\dot{\mathbf{x}}, \mathbf{r})$ projects the robot velocity onto the direction vector pointing from the obstacle to the robot and captures how much the robot moves towards the obstacle. Finally, the full obstacle avoidance policy is defined as the tuple $\mathcal{P}_{obs} = (\mathbf{f}_{obs}, \mathbf{A}_{obs})$:

$$\mathbf{f}_{obs}(\mathbf{x}, \dot{\mathbf{x}}, \mathbf{r}, d) = \mathbf{f}_{rep}(\mathbf{x}, \mathbf{r}, d) - \mathbf{f}_{damp}(\dot{\mathbf{x}}, \mathbf{r}, d) \quad (4)$$

$$\mathbf{A}_{obs}(\mathbf{x}, \dot{\mathbf{x}}, \mathbf{r}, d) = w_r(d,) \cdot \boldsymbol{s}(\mathbf{f}_{obs}) \boldsymbol{s}(\mathbf{f}_{obs})^T. \quad (5)$$

$\boldsymbol{s}(\cdot)$ is a soft-normalization function. Please refer to [6] for the detailed formulations of $\mathbf{P}_{obs}$ (eq. 68) and $\boldsymbol{s}$ (eq. 24). w_r scales the policy response based on a distance parameter r, which influences the policy's maximum active range according to $w_r(d) = \frac{1}{r^2}d^2 - \frac{2}{r}d + 1$. For each of the thousands of found obstacle cells such a policy is created. All cells at the same scale level λ are then summed according to eq. (1), and all resulting combined policies of all scales are then again summed using eq. (1). The scale level λ is used to set the RMP's parameters as follows: $v_{damp} = 0.45\lambda$, $v_{rep} = 0.75\lambda$, and $r = 1.5\lambda$. Modulating v_{damp}, v_{rep}, and r, allows setting the sphere of influence of policies, and for example determines the traversability of narrow corridors. By using the tuning proposed above, coarse obstacles naturally have a larger sphere of influence. The distance and size of the
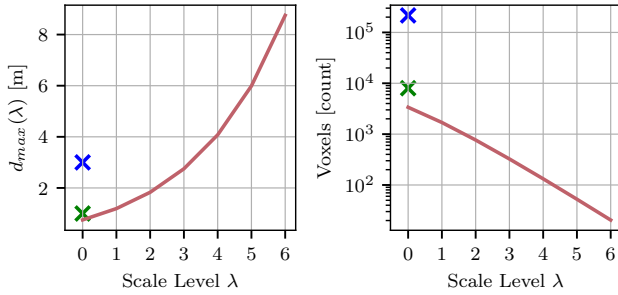
Fig. 4. Left: Perceptive radius defined by $d_{max}(\lambda)$ as used in the obstacle filter (red). Limited, fixed-resolution comparison variants used in V-A are marked with a blue resp. green cross. Right: Worst-case counts of voxels to visit. Even with small perceptive radii, the fixed-resolution variants need to potentially iterate over significantly more voxels to provide the same quality of obstacle avoidance (log-scale).

obstacle cell are used to scale the policy's Riemannian metric, which can be interpreted as a multi-dimensional weight and modulates the policy's strength and activation radius. The Riemannian metric ensures that the relative direction to the obstacle cell is taken into account such that there is only a repulsion component if the robot's velocity points towards this obstacle. In Figure 3, examples of obstacle cells are shown for both uniform and hierarchical cell generation.

D. Navigation system integration

We use the simple goal-attractor policy described in [6] to combine the previously described summation of obstacle avoidance policies with goal-seeking behavior. The goal-attractor policy is defined as:

$$\mathbf{f}_a(\mathbf{x}, \dot{\mathbf{x}}) = \alpha_a \boldsymbol{s}(\mathbf{x}_a - \mathbf{x}) - \beta_a \dot{\mathbf{x}},$$
$$\mathbf{A}_a(\mathbf{x}, \dot{\mathbf{x}}) = \mathbb{I}^{3 \times 3} \qquad (6)$$

where $\alpha_a, \beta_a > 0$ are tuning parameters, and $\mathbf{x}_a$ is the desired goal location. In each iteration, all policies are evaluated, summed up, and the resulting acceleration executed on the robot. For simulation experiments, the policies are run as fast as possible, whereas during field tests the policies are evaluated at the robot's control frequency (200 Hz). Note that it is straightforward to replace or combine the goal-seeking policy with other policies addressing tasks such as visual servoing, terrain following, manipulation, or assisted manual control, as has been shown e.g. in [22].

IV. HIERARCHICAL POLICY APPROXIMATION ERROR

Naturally, one wonders what the impact of incorporating distant obstacles at a reduced resolution is. In this section, we study the influence of replacing a sum of obstacle avoidance policies with a single policy at the center of such a block. In the obstacle cell extraction algorithm, the octree is traversed to a deeper or shallower level depending on the distance to the robot. This implies that at larger distances, fewer policies at slightly different locations contribute to the overall navigation result instead of a sum of many individual policies. In the following, we show what relative changes in policy outputs and quality these abstractions entail, using the toy example in Figure 5 for the analysis. We conduct a numerical analysis to simulate the relative changes between
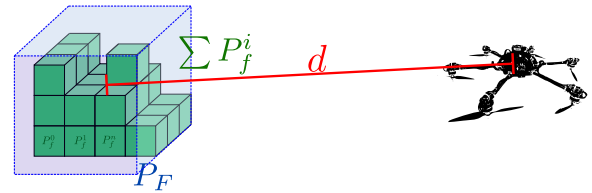

Fig. 5. Example of obstacles that can be either modeled by a single, large policy (P_F) or multiple small, high-resolution policies (P_f^i). The distance d represents the distance from the robot to the center of the obstacle block.
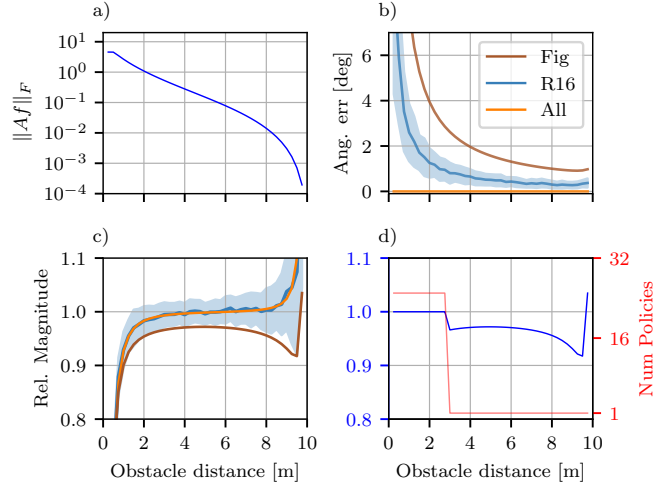

Fig. 6. a) Plot of the typical absolute policy strength (log-plot) w.r.t. obstacle distance. Subfigure b) and c) visualize the angular and relative magnitude error of approximating the high-resolution policies with a single coarse approximation. 'Fig' shows this for the exact configuration seen in fig. 5, 'R16' for a random selection of 16 occupied voxels at high resolution (thus the covariance), and 'All' for a fully occupied volume. d) Illustration of the approximation error for a hierarchical policy, where a full-resolution policy is used below 2.5 m distance and a single summary policy at larger distances. The spike in the approximation error's magnitude at a distance of 10 m is unimportant, as the absolute strength at this distance nears 0.

the single simplified policy P_F and the granular, high-resolution set of policies $\sum P_f^i$ in both policy strength and directionality for three scenarios: 1) the toy example in Figure 5 (labeled "Fig" in Figure 6), 2) a random sampling of 16 occupied voxels, respectively their resulting policies ("R16"), and 3) a completely occupied block resulting in 64 policies ("All"). The same $4 \times 4 \times 4$ block with 10 cm voxels is used in all scenarios. As is visible in Figure 6, the induced errors are negligible both in angular error as well as relative strength (magnitude) of the resulting policies. As to be expected, errors are higher when the distance to the voxels is smaller. The combination of multiple policies at different scales provides the best compromise; it minimizes the number of policies needed while also providing low approximation error over the entire distance.

V. EXPERIMENTS

We perform a comprehensive set of experiments to evaluate the navigation success rates, computational efficiency, and real-world applicability of the proposed system. To provide context, we include comparisons with CHOMP [8]. CHOMP generates complete trajectories and requires an Euclidean Distance Field (EDF), which is time-consuming to
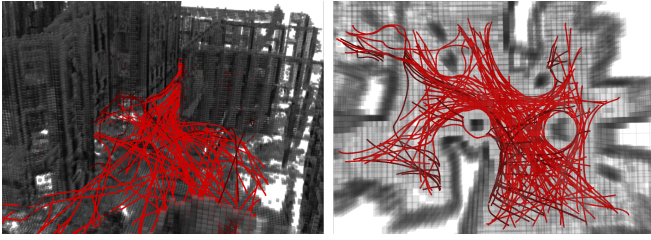
Fig. 7. Qualitative visualization of the map scenarios used for statistical evaluation. Left shows a perspective rendering of the `math` scenario, right is a top-down rendering of the `mine` scenario. The red lines are example trajectories from our proposed navigation algorithm. Trajectories stuck in local minima are marked in dark red.

generate ($\approx 30\,\text{s}$ for the maps used). By contrast, an RMP-based navigation framework is inherently reactive and only needs obstacle information which is readily available in the volumetric map.

A. Statistical evaluation and comparison

Despite the purely reactive nature of the proposed system, we are interested in its capability and performance in finding moderately complex trajectories in realistic maps. To this end, we perform an in-depth randomized evaluation on maps generated from the *Newer College* LiDAR dataset [23] with a min voxel size of $10\,\text{cm}$. We use subsections of two maps – `mine` and `math`, visualized in Figure 7 – in which we sample random start and end points and let each navigation algorithm find a smooth, collision-free trajectory. We evaluate a total of four algorithms; a) the proposed hierarchical system as described in section III, b) an implementation of CHOMP [8], c) a non-hierarchical variant of our system that only uses the highest resolution voxels, up to a maximum distance of $1\,\text{m}$, and d) $3\,\text{m}$, respectively. The non-hierarchical variants serve to illustrate the effects of the reduced perceptive radius, which is limited due to significantly increased compute costs inherent to single-resolution approaches at small voxel sizes. All reactive, RMP-based variants are used in an end-to-end fashion, meaning that the policies are repeatedly updated and integrated until the robot is at a stand-still, either at the goal or in a local minima. Obstacle cells are updated from the map whenever the displacement since the last update exceeds $0.05\,\text{m}$. CHOMP is configured to run with $N = 500$ trajectory points until it converges ($\epsilon_{rel} < 1e^{-5}$) or a maximum iteration count (100) is reached. To demonstrate the relative performance of the proposed system, we provide a detailed look at planning success rate, planning time, and distances to obstacles. Figure 8 shows the relative amount of successfully found trajectories, i.e. that reach the goal location and do not get stuck. All algorithms perform similarly well and solve about 75% of all tasks, which is rather good considering that they are all local and not global planners. Due to their different nature, the reactive algorithms get (safely) stuck in local minima, whereas the optimization-based CHOMP method may simply not converge to a solution that is collision free. Figure 9 provides detailed statistics of the measured run-times of the different algorithms. Noteworthy is the drastic increase in
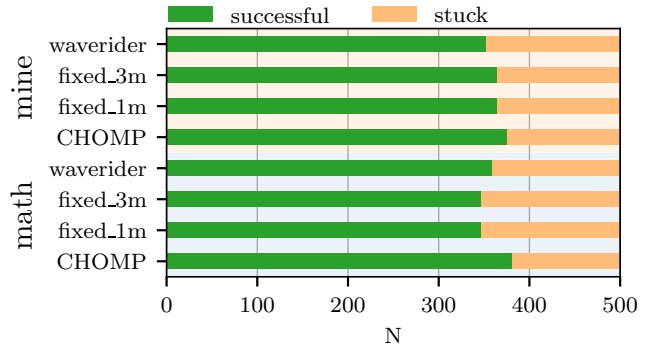

Fig. 8. Success rates for all algorithms on both maps with 500 randomized trials each. CHOMP runs that did not terminate within the allocated time budget are labeled as stuck.
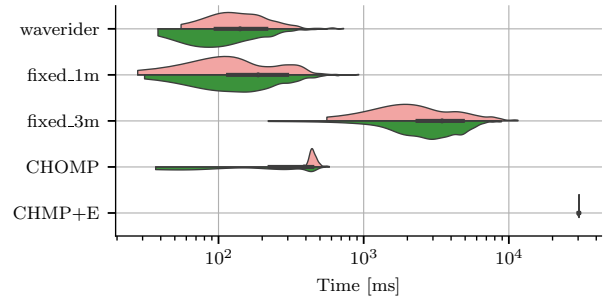

Fig. 9. Timing distributions for rest-to-rest trajectories on map `math`. Green parts only include successful trajectories, red parts only stuck ones. CHOMP clearly shows increased calculation time for failing trajectories as it runs more solver iterations. Note the log scale and the drastically increased run-time for the fixed-resolution variant. For context, `CHMP+E` visualizes the cost of a trajectory, including the necessary collision distance (EDF) pre-processing for a map for CHOMP.

run-time with larger perceptive radii, which makes the use of large amounts of occupancy information intractable when a fixed-resolution representation is used. A major difference between our proposed method and CHOMP is its pure reactive nature. While we compare full rest-to-rest trajectory run-times in Figure 9, in practice only a single iteration is calculated at every controller iteration. Effectively, this provides full obstacle avoidance navigation at a marginal compute cost – approximately $100\,\mu\text{s}$ per step on average – and a few milliseconds per step involving obstacle updates. Conversely, CHOMP only provides results after full convergence. To provide insights into trajectory safety, we evaluate the distance to the closest occupied obstacle for each step along each evaluated trajectory from the randomized tests. The resulting distributions are visualized as histograms in Figure 10. The proposed hierarchical approach shows a safe distribution with no parts of the trajectories getting close to obstacles. The two fixed-resolution algorithms frequently travel *much* closer to obstacles due to their limited perceptive fields, whereas CHOMP may output unsafe states in case of non-convergence. Finally, Figure 1 shows a visualization of trajectories generated in four example scenarios. These examples show that both fixed-resolution variants produce poor and unsteady trajectories due to their limited perceptive range. Combining all the presented results, the proposed multi-resolution, purely reactive, hierarchical algorithm pro-
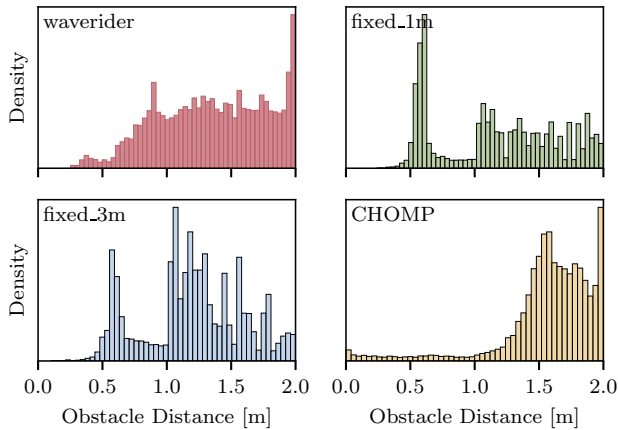
Fig. 10. Histogramms of distance to obstacles over 50 000 random trajectory traces from each algorithm. The EDF used in the evaluation is truncated at 2 m, with everything above that value being considered far away from obstacles.

vides an attractive compromise of run-time, success rate, and trajectory quality at marginal compute cost.

B. Computational efficiency

We benchmark the computational efficiency of the proposed navigation system on a NVidia Jetson Orin AGX computer, using data from a Livox Mid-360 LiDAR. The navigation algorithm only uses the computer's 12-core ARM Cortex-A78AE CPU. Figure 11 visualizes the latency and policy processing times on a dataset that traverses indoor offices before transitioning to a terrace, including a large 30 m radius open space. Together, the mapping and planning use 2.4 CPU threads (average) and 355 MB of RAM (peak). The LiDAR delivers new data every 100 ms and integrating these observations takes 29 ms (average), while selecting and executing the obstacle avoidance policies takes 6.9 ms (average). All together, the mapping and planning steps are completed almost instantaneously after the LiDAR data is received.

C. Platform tests

The proposed navigation pipeline (Figure 2) is deployed on an Micro Aerial Vehicle (MAV) with a Livox Mid-360 LiDAR for odometry [24] and mapping. We run the aerial robot through an indoor obstacle course without a prior map, such that all data used for navigation must be gathered and processed on the fly. The operator sets a desired goal location prior to the flight, which the robot then autonomously tries to reach using the proposed reactive navigation algorithm. Figure 12 visualizes a typical path taken by the aerial robot to avoid an obstacle and fly towards a (potentially unreachable) goal position. Upon setting a desired goal position, the goal-seeking policy starts to drive the robot. After about 130 ms, the first scan is received, the map is populated and the obstacle avoidance policies become active. As can be seen from Figure 12, the robot avoids the obstacles with sufficient distance. During the full run, the robot never got closer than 0.75 m to an obstacle and kept an average closest-obstacle distance of 1.16 m ± 0.32 m.
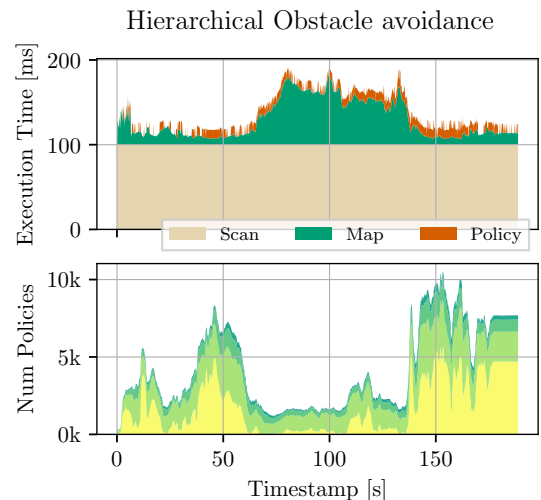

Fig. 11. Top: Stackplot of data latency (LiDAR) and processing latency (Map/Policy). Bottom: Visualization of the number of policies at different levels, where yellow is the finest resolution and dark green is the coarsest, in similar fashion to Figure 3. The system is on the outdoor terrace between 55 s − 130 s. Especially after the robot reenters the building, it is in close proximity to many obstacles, leading to more policies at a higher resolution.
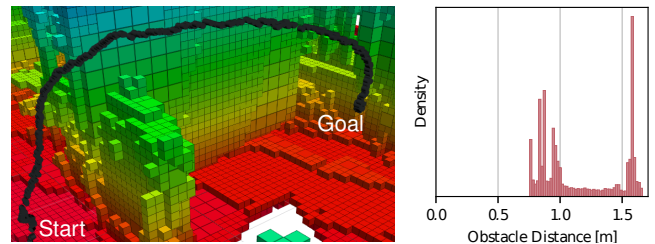

Fig. 12. Left: Rendering of an executed flight path (black) and the map that was created during a traversal of a cluttered region with the specified goal location. Right: Obstacle distance histogram for the same flight. The MAV successfully cleared all obstacles with sufficient margin. Note: For operational reasons, the tuning for the field test was more conservative (stronger) than for the map-based evaluation.

VI. CONCLUSION

In this paper, we presented a novel method for multi-resolution, reactive obstacle avoidance in generic 3D environments. A key insight is the efficient use of multi-resolution, hierarchical obstacle information. This follows the intuition that geometry further away does not need to be incorporated at the same resolution as nearby obstacles. As demonstrated through numerical analysis and ablations, the proposed approach enables locally precise and safe collision avoidance while keeping a very large perceptive radius. Multi-resolution obstacles can efficiently be extracted by directly exploiting the hierarchical structure present in hierarchical volumetric mapping frameworks such as wavemap [5]. The proposed system achieves planning success rates comparable to CHOMP while reducing the planning time by 50× and requiring no pre-processing or post-processing steps, such as EDF generation and trajectory tracking control. Finally, the system is deployed on a real MAV negotiating an indoor obstacle course while only using minimal computational resources.

REFERENCES

[1] H. Oleynikova, D. Honegger, and M. Pollefeys, "Reactive avoidance using embedded stereo vision for mav flight," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2015, pp. 50–56.

[2] A. Hornung, K. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous Robots*, 2013.

[3] D. Duberg and P. Jensfelt, "Ufomap: An efficient probabilistic 3d mapping framework that embraces the unknown," *IEEE Robotics and Automation Letters*, 2020.

[4] E. Vespa, N. Nikolov, M. Grimm, L. Nardi, P. H. J. Kelly, and S. Leutenegger, "Efficient octree-based volumetric SLAM supporting signed-distance and occupancy mapping," *IEEE Robotics and Automation Letters*, 2018.

[5] V. Reijgwart, C. Cadena, R. Siegwart, and L. Ott, "Efficient volumetric mapping of multi-scale environments using wavelet-based compression," in *Robotics: Science and Systems*, 2023.

[6] N. D. Ratliff, J. Issac, D. Kappler, S. Birchfield, and D. Fox, "Riemannian Motion Policies," *arXiv:1801.02854 [cs]*, 2018.

[7] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Autonomous robot vehicles*, Springer, 1986, pp. 396–404.

[8] M. Zucker *et al.*, "Chomp: Covariant hamiltonian optimization for motion planning," *International Journal of Robotics Research*, 2013.

[9] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017.

[10] L. Han, F. Gao, B. Zhou, and S. Shen, "Fiesta: Fast incremental euclidean distance fields for online motion planning of aerial robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2019.

[11] K. Mohta *et al.*, "Fast, autonomous flight in gps-denied and cluttered environments," *Journal of Field Robotics*, 2018.

[12] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *International Journal of Robotics Research*, 2011.

[13] H. Oleynikova *et al.*, "An open-source system for vision-based micro-aerial vehicle mapping, planning, and flight in cluttered environments," *Journal of Field Robotics*, 2020.

[14] A. Loquercio, E. Kaufmann, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza, "Learning high-speed flight in the wild," *Science Robotics*, vol. 6, no. 59, eabg5810, 2021.

[15] M. Pantic *et al.*, "Obstacle avoidance using raycasting and riemannian motion policies at khz rates for mavs," in *IEEE International Conference on Robotics & Automation*, 2023.

[16] S. Kambhampati and L. Davis, "Multiresolution path planning for mobile robots," *IEEE Journal on Robotics and Automation*, vol. 2, no. 3, pp. 135–145, 1986. DOI: 10.1109/JRA.1986.1087051.

[17] W. Du, F. Islam, and M. Likhachev, "Multi-resolution A*," *ArXiv*, vol. abs/2004.06684, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID: 215754159.

[18] N. Funk, J. Tarrio, S. Papatheodorou, M. Popović, P. F. Alcantarilla, and S. Leutenegger, "Multi-resolution 3d mapping with explicit free space representation for fast and accurate mobile robot motion planning," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3553–3560, 2021. DOI: 10.1109/LRA.2021.3061989.

[19] D. M. Saxena, T. Kusnur, and M. Likhachev, "AMRA*: Anytime multi-resolution multi-heuristic A*," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 3371–3377. DOI: 10.1109/ICRA46639.2022.9812359.

[20] N. Funk, J. Tarrio, S. Papatheodorou, P. F. Alcantarilla, and S. Leutenegger, "Orientation-aware hierarchical, adaptive-resolution A* algorithm for UAV trajectory planning," *IEEE Robotics and Automation Letters*, vol. 8, no. 10, pp. 6723–6730, 2023. DOI: 10.1109/LRA.2023.3308490.

[21] K. Goel, Y. G. Daoud, N. Michael, and W. Tabib, "Hierarchical collision avoidance for adaptive-speed multirotor teleoperation," in *2022 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 2022, pp. 20–27. DOI: 10.1109/SSRR56537.2022.10018782.

[22] M. Mattamala, N. Chebrolu, and M. Fallon, "An efficient locally reactive controller for safe navigation in visual teach and repeat missions," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2353–2360, 2022. DOI: 10.1109/LRA.2022.3143196.

[23] L. Zhang, M. Camurri, D. Wisth, and M. Fallon, "Multi-camera lidar inertial extension to the newer college dataset," *arXiv preprint arXiv:2112.08854*, 2021.

[24] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, "Fast-lio2: Fast direct lidar-inertial odometry," *IEEE Transactions on Robotics*, 2022.