Search-based Planning of Dynamic MAV Trajectories Using Local Multiresolution State Lattices

Daniel Schleich and Sven Behnke

Abstract— **Search-based methods that use motion primitives can incorporate the system's dynamics into the planning and thus generate dynamically feasible MAV trajectories that are globally optimal. However, searching high-dimensional state lattices is computationally expensive. Local multiresolution is a commonly used method to accelerate spatial path planning. While paths within the vicinity of the robot are represented at high resolution, the representation gets coarser for more distant parts. In this work, we apply the concept of local multiresolution to high-dimensional state lattices that include velocities and accelerations. Experiments show that our proposed approach significantly reduces planning times. Thus, it increases the applicability to large dynamic environments, where frequent replanning is necessary.**

I. INTRODUCTION

The application of micro aerial vehicles (MAVs) to surveillance, inspection, and search & rescue tasks has gained increasing popularity in recent years. Generating fast, dynamically feasible MAV trajectories for such scenarios is a challenging task, since one usually has to deal with large, initially unknown environments. Fast trajectory replanning is necessary to avoid dynamic or previously unknown obstacles. Many existing methods generate trajectories in a two-stage approach: First, a 3D position-only obstacle-free path is planned using search-based or sampling-based methods [1], [2]. Afterwards, this path is refined to a high-dimensional trajectory including velocities. Refinement strategies include quadratic programming [3], B-Spline path planning [4] and gradient-based optimization methods [5] as used in [6]. These approaches only generate locally optimal trajectories. However, incorporating the system's dynamics into the planning process allows to generate globally optimal trajectories. Liu et al. [7] propose a search-based method to directly plan second- and third-order MAV trajectories. They unroll motion primitives to generate a state lattice graph, from which they extract trajectories using A* with a heuristic based on the solution of a Linear Quadratic Minimum Time problem. Since velocities are explicitly considered during planning, their approach can be extended to generate trajectories through doors that are narrower than the MAV diameter [8]. However, searching high-dimensional state spaces is computationally expensive. Thus, such approaches either lack the capability for fast replanning or are restricted to small or low-resolutional environment representations.
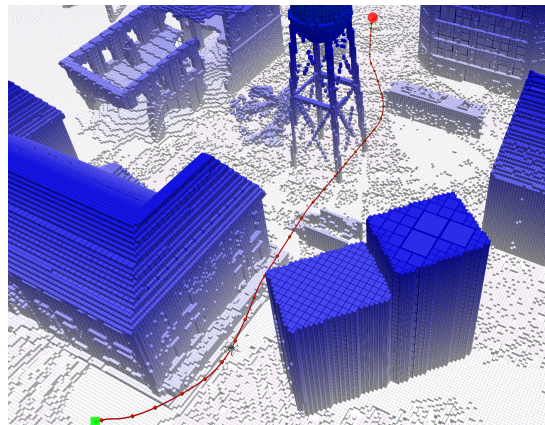
Fig. 1: The MAV following a trajectory in a simulated outdoor environment. The start is at the red circle and the goal is marked by the green square. The trajectory is replanned online at 1 Hz. Red cubes mark the start positions of each replanning step.

A common approach for reducing the state space size for spatial path planning is local multiresolution [9]. Here, only the vicinity of the robot is represented at high resolution, while the resolution decreases with increasing distance from the robot. In this work, we apply the concept of local multiresolution to the state lattice graph of Liu et al. [7]. Thus, we are able to significantly reduce planning times. This increases the applicability of the approach to large dynamic environments, where frequent replanning is necessary. In summary, the main contributions of this paper are:

- The introduction of local multiresolutional state lattices,
- an expansion scheme for the A* algorithm addressing the issues of multiresolutional state representations, and
- a search heuristic based on the solution of 1D sub-problems.

II. RELATED WORK

Multiresolutional approaches are commonly used to accelerate planning. Behnke [9] proposed A*-based path planning on robot-centered local multiresolution grids, where the spatial resolution decreases with increasing distance from the robot. Nieuwenhuisen et al. [10] extend this approach to plan 3D trajectories for MAVs. In [11], local multiresolution in time is used to enable fast reoptimization of an initial path from a grid-based planner. Du et al. [12] perform multiple A* searches simultaneously on grids with different resolutions while sharing states that lie on multiple resolution grids.

Multiresolution has also been used in combination with state lattices. González-Sieira et al. [13] choose the resolution level based on the complexity of the local environment, i.e.,

on the distance from obstacles. They apply different sets of motion primitives for each level. In [14], the same authors group motion primitives in categories. When expanding a state, the longest collision-free motion from each category is applied. Pivtoraiko et al. [15] use a uniform 2D grid but apply different state transitions for start and goal area than for the other areas. This framework is extended by Andersson et al. [16]. They propose a receding-horizon method, where the first half of the planning time is used for planning in a high-dimensional space. The search is continued afterwards with lower dimension and a reduced action set. All of the above state lattice methods use different action sets dependent on the current representation level. However, all levels share the same spatial resolutions. In contrast to those methods, our approach reduces the spatial resolution with increasing distance from the current MAV position.

Likhachev et al. [17] propose multi-resolutional state lattices to plan trajectories for autonomous ground vehicles. The action set of the lower-resolution level is a strict subset generated from the higher-level action set by choosing only actions whose end states lie on the low-dimensional grid. The state representation used by Likhachev et al. only includes movement directions, but their approach is expanded by Rufli et al. [18] to also consider velocities. Petereit et al. [19] use two representation levels with different dimensionality. The representation level of a state is chosen based on the time it takes to reach this state. Similar to the method of Likhachev et al., transitions from higher to lower representation levels are only possible for states that match the resolution of the lower level.

Our method uses the same dimensionality for all representation levels but reduces spatial and velocity resolutions for higher levels. In contrast to the approaches mentioned above, we do not rely on a fixed precomputed set of motion primitives. For transitions between different representation levels, we adjust the motion primitives such that the resulting states match the resolution of the target level.

III. METHOD

Our trajectory planning method is based on the framework of Liu et al. [7]. In this work, we restrict ourselves to second-order systems and leave the extensions to third-order systems for future work. Thus, we model the MAV state as a 6-tuple $s = (\mathbf{p}, \mathbf{v}) \in \mathbb{R}^6$ consisting of a 3D position $\mathbf{p}$ and velocity $\mathbf{v}$. Mind, that we do not explicitly model the MAV yaw, since it does not influence the system's dynamics. Instead, we choose the yaw in a post-processing step based on the flight direction of the planned trajectory.

The state space is discretized by unrolling motion primitives from the initial MAV state $s_0 = (\mathbf{p_0}, \mathbf{v_0})$, resulting in a state lattice graph $\mathcal{G}(\mathcal{S}, \mathcal{E})$. Here, $\mathcal{S}$ denotes the set of discretized MAV states, and $\mathcal{E}$ is the set of motion primitives connecting the states. Each motion primitive is generated by applying a constant acceleration $\mathbf{u}$ from a discrete control set $\mathcal{U}_M \subset \mathbb{R}^3$ over a short time interval τ. Thus, the motion primitive $F_{\mathbf{u},s}$ connecting the states s to $s' := F_{\mathbf{u},s}(\tau)$ can
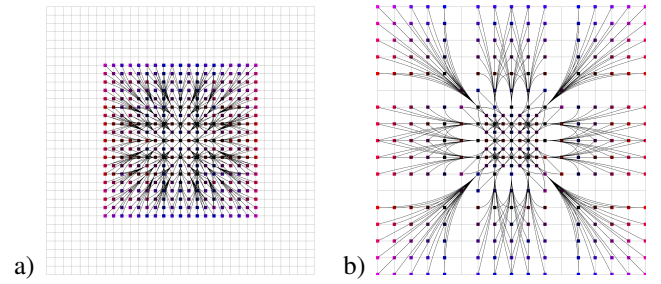

Fig. 2: Top-down view of 2D state lattice graphs. a) Uniform. b) Local multiresolution. The spatial position of nodes is fixed to the corners of a multiresolution grid with high resolution at the center, i.e., the current MAV position, and coarser resolution for more distant areas. Bright red represents high absolute velocity along the x-axis, bright blue represents high velocity along the y-axis.

be expressed as a time-parameterized polynomial

$$F_{\mathbf{u},s}(t) = \begin{pmatrix} \mathbf{p} + t\mathbf{v} + \frac{t^2}{2}\mathbf{u} \\ \mathbf{v} + t\mathbf{u} \end{pmatrix}, \text{ for } t \in [0, \tau]. \quad (1)$$

An example for the resulting state lattice graph is depicted in Fig. 2 a.

The costs for a motion primitive are defined as the weighted sum of control effort and primitive duration, i.e.,

$$C(F_{\mathbf{u},s}) = ||\mathbf{u}||_2^2 \tau + \rho\tau. \quad (2)$$

The optimal trajectory can be planned in the state lattice graph $\mathcal{G}$ by applying graph search methods like A*. For more details, we refer to [7].

In the following, we introduce multiple concepts to reduce planning times. In Sec. III-A, we reduce the state space size by applying the concept of local multiresolution to the state lattice graph $\mathcal{G}$. The idea is to restrict the state positions to the corners of a MAV-centered local multiresolution grid, as shown in Fig. 2 b. Additionally, we reduce the number of discrete velocities for states whose positions are far from the current MAV position.

In Sec. III-B, we propose a search heuristic based on the solution of 1D sub-problems. To avoid overshooting the goal position, we introduce special goal action sequences in Sec. III-C. Finally, Sec. III-D details an expansion scheme for the A* algorithm, specially designed to overcome issues of multiresolutional state representations.

A. Local Multiresolutional State Lattices

We define multiple levels of state discretization, dependent on the distance to the initial MAV position $\mathbf{p_0}$. *Level-1* covers all states whose positions are close to $\mathbf{p_0}$. The resolution represents the smallest possible position and velocity changes $\Delta_p^{(1)}$ and $\Delta_v^{(1)}$ of a motion primitive. They depend on the primitive duration τ and the minimal non-zero acceleration command $u_{\min}$, and they can be obtained from (1):

$$\Delta_p^{(1)} = \frac{1}{2}\tau^2 u_{\min}, \ \Delta_v^{(1)} = \tau u_{\min}. \quad (3)$$

The spatial resolutions of the higher levels are obtained by halving the resolution of the next lower level. For *Level-i*, we get $\Delta_p^{(i)} := 2^{i-1}\Delta_p^{(1)}$. If we choose $\tau < 1$, the velocity
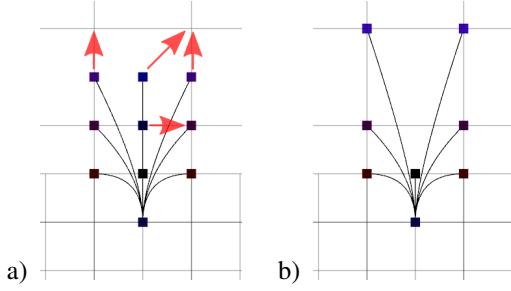
Fig. 3: Adjusting motion primitives to the local multiresolution grid. First, the lower-level motion primitives are unrolled (a). Then, the closest grid corners to the primitive end positions are determined (red arrows). Finally, primitives ending at those grid corners are generated (b).

resolution is much coarser than the spatial resolution. Therefore, we only halve the velocity resolution after every second resolution level, i.e., we define

$$\Delta_v^{(2)} := \Delta_v^{(1)}, \text{ and } \Delta_v^{(3)} := \Delta_v^{(4)} := 2\Delta_v^{(1)}. \qquad (4)$$

Each resolution level covers an eight times larger volume than the next lower level. They are embedded into each other such that the center of a higher level is replaced by the next lower level. Fig. 2 b shows an example with two levels.

When generating the state lattice graph by unrolling motion primitives, we have to ensure that the target states of the motion primitives match the discretization of the corresponding resolution level. This is done by carefully defining control sets $\mathcal{U}^{(i)}$ and time steps $\tau^{(i)}$ for each level. Mind that we define control sets independently for each spatial dimension. With respect to the choice of the time steps, there are two different approaches, which we explain in the following and evaluate against each other in Sec. IV.

a) Fixed time steps per level: From (3), it follows that the velocity resolution is halved when doubling the time step, while the spatial resolution is reduced by a factor of four. Thus, by choosing $\tau^{(i)} := 2^{i-1}\tau^{(1)}$, we ensure that position and velocity changes are multiples of the *Level-i* resolutions. If we additionally halve the acceleration commands, the velocity resolution does not change. Thus, for each command $u^{(1)} \in \mathcal{U}^{(1)}$, we define the corresponding higher-level commands as

$$u^{(2)} := u^{(3)} := \frac{1}{2}u^{(1)} \text{ and } u^{(4)} := \frac{1}{4}u^{(1)}. \qquad (5)$$

Halving the commands while doubling durations significantly reduces maneuverability, however. We mitigate this effect by adding two special actions to each level: decelerate to zero velocity and accelerate to the maximum allowed velocity in current flight direction.

The above method enforces that end states of all motion primitives starting in *Level-i* also lie on the *Level-i* grid. However, level transitions have to be specially addressed (see Fig. 3): We predict the end state of a motion primitive, determine the closest position $\mathbf{p_g}$ on the target resolution grid, and choose the acceleration command $\mathbf{u}$ such that the primitive ends at $\mathbf{p_g}$:

$$\mathbf{u} = 2\frac{\mathbf{p_g} - \mathbf{p} - \tau^{(i)}\mathbf{v}}{(\tau^{(i)})^2}, \qquad (6)$$

where $(\mathbf{p}, \mathbf{v})$ denotes the start state of the motion primitive. Using $\Delta_v = \tau^{(i)}\mathbf{u}$, it follows that the resulting velocity change always is a multiple of the *Level-1* velocity resolution, but for higher levels, the velocity might not match the coarser resolution. Intermediately, we allow this offset for the end states of the adjusted primitives and correct it only when generating their successors.

b) Variable time steps per level: Position changes do not only depend on τ and $u_{\min}$ but also on the initial velocity. Using large time steps for high velocities results in a much lower spatial resolution. Thus, many motion primitives with high initial velocities become invalid in the presence of obstacles. Valid trajectories can still be found by choosing lower velocities. However, this increases the trajectory costs, which leads to a significant increase of node expansions during the search (compare Sec. IV-B). Thus, it might be a good idea to choose the time steps dependent on the current velocity. For each command u and initial velocity v, we choose the smallest time step $\tau^{(i)} \in \{2^k\tau^{(1)}, k \in \mathbb{N}_0\}$, such that the position change $\tau^{(i)}v + \frac{1}{2}(\tau^{(i)})^2 u$ is larger than the spatial resolution $\Delta_p^{(i)}$ of the current level. If the position does not change, we set $\tau^{(i)} = \tau^{(1)}$. Mind that we use the same command set for each resolution level. Additionally, we adjust the motion primitive end points to the multiresolution grid as described above.

B. 1D Heuristic

Liu et al. [7] use a heuristic based on the solution of a Linear Quadratic Minimum Time problem. Their heuristic considers constraints on the maximum velocity but assumes continuous instead of piecewise-constant control commands. Furthermore, the commands are only implicitly bounded by optimizing the control effort and thus might violate constraints on the maximum acceleration.

We propose a heuristic based on precomputing the actual costs for a 1D problem. To reduce the size of the lookup table, we assume that all goal states have zero velocity. For each pair of signed distance to the goal position and start velocity, we precompute the costs of the optimal 1D trajectory using the *Level-1* resolutions.

The costs of a trajectory $\mathbf{u}_{0:N-1}$ of length N and duration $T = \sum_{k=0}^{N-1} \tau_k$ can be approximated by

$$\begin{aligned} C(\mathbf{u}_{0:N-1}) &= \rho T + \sum_{k=0}^{N-1} ||\mathbf{u}_k||_2^2 \tau_k \\ &= \rho T + \sum_{k=0}^{N-1} (u_k)_x^2 \tau_k + \sum_{k=0}^{N-1} (u_k)_y^2 \tau_k + \sum_{k=0}^{N-1} (u_k)_z^2 \tau_k \\ &>= \rho T + c_x + c_y + c_z, \end{aligned} \qquad (7)$$

where c_x, c_y, c_z are lower bounds on the control efforts along the individual dimensions.

During search, we look up the times $T_x^{1D}, T_y^{1D}, T_z^{1D}$ and control efforts $c_x^{1D}, c_y^{1D}, c_z^{1D}$ of the corresponding 1D subproblems. The total time T of the 3D trajectory is given by $T = \max\{T_x^{1D}, T_y^{1D}, T_z^{1D}\}$. Without loss of generality, let $T = T_x^{1D}$. Thus, we can set $c_x = c_x^{1D}$. However, c_y^{1D} and

c_z^{1D} might overestimate c_y and c_z if the 1D trajectories have lower durations than the 3D trajectory. Therefore, we choose c_y and c_z dependent on the current flight direction along the corresponding dimensions:

- When flying towards the goal, we choose the control costs that correspond to a full stop.
- If the velocity is zero, we choose the control costs that correspond to applying minimal acceleration u_{min} followed by deceleration $-u_{min}$.
- When flying away from the goal, the MAV has to stop, accelerate towards the goal, and decelerate again. Thus, the resulting control cost is the sum of the above cases.

Note that the 1D heuristic is admissible but not consistent: If the dimension, for which the maximum time is achieved, changes, the control costs of a different 1D sub-problem will be used. Thus, the decrease of the estimated 3D control costs might be larger than the costs of the applied action.

C. Goal Actions

When increasing the motion primitive duration for higher-resolution levels, we might overshoot the goal position frequently during search. Thus, when expanding a state s, we check whether the goal is reachable: Let s be represented within *Level-i*. We determine lower and upper bounds $p_{min}, p_{max} \in \mathbb{R}^3$ of the area that the MAV can reach within the next time step. Here, we assume a possible acceleration of $\pm u_{max}$ for a duration of $2^{i-1}\tau^{(1)}$ along each dimension. If the goal position lies within $[p_{min}, p_{max}]$, we check whether it can be reached from s using a sequence of *Level-1* motion primitives with a total duration of at most $2^{i-1}\tau^{(1)}$. If so, we add the goal state as a neighbor of s, connected by the resulting motion primitive sequence. As for the heuristic, we do not consider any obstacles and assume that the goal state has zero velocity. Thus, the sequences of *Level-1* motion primitives can be efficiently precomputed for 1D sub-problems and looked up based on the current velocity and distance to the goal.

D. Level-Based Expansion Scheme

The state space size is significantly reduced when using local multiresolution. However, experiments show that the A* algorithm tends to expand much more states when applied to multiresolutional state lattices compared to uniform lattices. We found that a reason for this behavior is the fact that the A* algorithm expands all states whose estimated costs are lower than the costs of the optimal solution (see Sec. IV-B). Thus, we adapt the A* algorithm such that it does not always expand the state with the lowest estimated costs, i.e., lowest f-value, but also might expand states with higher values. This might result in sub-optimal solutions, which we accept since the multiresolutional lattice representation already introduces sub-optimality.

For each resolution level, we use a separate priority queue. To determine which node to expand, we compare the state with lowest f-value from each level. A state is considered an expansion candidate if the difference between its f-value and the globally minimal value is at most the cost of one step
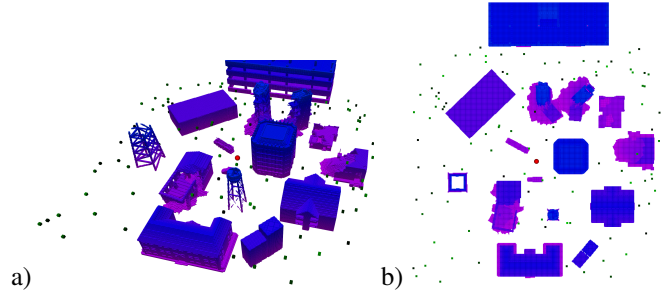

Fig. 4: The evaluation environment. a) 3D. b) Top-down. The start position is at the map center (red sphere). Goal positions are marked with squares which are brighter for larger height values.

in the corresponding resolution level. From all candidates, we expand the one with the lowest heuristic value. We stop the search as soon as the goal state is added to the *OPEN*-list, instead of waiting until the goal state is expanded. This further reduces the runtime at the cost of possible suboptimality.

IV. EVALUATION

We use the following parameters for uniform planning as well as for *Level-1* of multiresolutional planning:

ρ	τ	v_{max}	u_{max}	du
16	0.5 s	4 m/s	2 m/s^2	2 m/s^2

.

The control set $\mathcal{U}^{(1)}$ for each dimension is obtained by discretizing $[-u_{max}, u_{max}]$ uniformly with resolution du. This results in *Level-1* resolutions of $\Delta_p^{(1)} = 0.25$ m and $\Delta_v^{(1)} = 1$ m/s. The time cost factor $\rho = 4u_{max}^2$ was chosen as proposed by Liu et al. [8].

We evaluate the proposed methods in a simulated outdoor environment of size 128×128 m containing several buildings (see Fig. 4). The allowed flight altitude is 0-10 m and positions which are closer to obstacles than 1.5 m are considered invalid.

All planning methods are applied to the same set of 100 trajectory generation tasks. The start position for all tasks is located at the map center with altitude 2 m. Goal positions are sampled uniformly at random with the additional constraint that the distance to the closest obstacle or the map border is at least the spatial resolution of the corresponding level.

We are interested in evaluating the ability for frequent replanning. Thus, after a trajectory is generated, we refine it by replanning from the MAV state that will be reached after one second. This process is repeated until the goal state is reached. For each planning task, we consider the maximum replanning time and expansion number over all replanning steps. Furthermore, we report the costs of the fully refined trajectories. If no solution is found within three million node expansions, we abort the planning process. If a solution is found, the average number of replanning steps lies between 13 and 18. Depending on the success rate, each method is thus applied to 900 to 1400 different planning tasks.
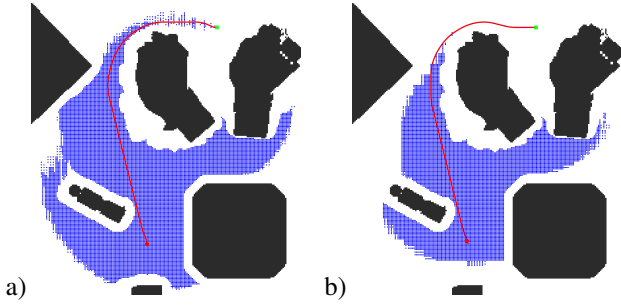
Fig. 5: Expanded nodes using different heuristics for a planning task in two spatial dimensions. a) Baseline Heuristic (115,988 expansions). b) 1D Heuristic (63,769 expansions). Positions of expanded nodes are marked with blue squares. The start position is marked with a red circle, the goal with a green square.

TABLE I: Comparison of the baseline heuristic h_{base} from [7] and our proposed 1D heuristic h_{1D}. For each planning method, the number of expansions and trajectory costs are averaged over all tasks where the corresponding method found valid trajectories with both heuristics. Note that the considered tasks might be different for different planning methods.

		Uniform	MRes$_{\text{fixed}}$	MRes$_{\text{variable}}$
h_{base}	Expansions	1,133,136	877,041	890,564
	Costs	**243.69**	**260.79**	**270.14**
h_{1D}	Expansions	**75,562**	**175,528**	**135,571**
	Costs	245.49	264.59	272.16

A. 1D Heuristic

We evaluate the effect of our proposed 1D Heuristic separately for the different planning methods (see Tab. I). The number of node expansions is significantly reduced for all three methods. The largest effect is achieved for uniform planning, where the number of expansions is reduced by 93%. For the multiresolutional planning methods, there is a reduction of 80% and 85%, respectively. Since the 1D heuristic is not consistent, the generated trajectories are not optimal anymore. However, the trajectory costs only increase about 0.75% for uniform planning and multiresolution with variable time steps. The increase for fixed level-dependent time steps is slightly larger with 1.46%. Fig. 5 visualizes the set of expanded nodes for a uniform planning task in 2D. Since our 1D heuristic significantly accelerates the A* search, we use it for all subsequent experiments.

B. Multiresolutional State Lattices with Standard A*

To evaluate the effect of multiresolutional state lattices, we record for each trajectory generation task the maximal replanning time and maximum number of node expansions. Tab. II reports the averages of these values over the tasks where all three planning methods generated valid trajectories. Interestingly, using multiresolutional state lattices with fixed level-dependent motion primitive durations significantly increases the maximum replanning time. Furthermore, maximum replanning times exceed 1 s more often for both multiresolutional state lattice variants. Although the size of the state space is reduced when using multiresolution, those methods tend to expand more states during A* search for some tasks.

TABLE II: Planning statistics for different state lattice representations. Maximum replanning times, number of expansions and trajectory costs are averaged over the tasks where all three planning methods found valid solutions. Additionally, the fraction of tasks for which the longest replanning step exceeds 1 s are given.

	Mean Maximum Replanning Time	Time > 1 s	Expansions	Costs
Uniform	3.54 s	**46.07%**	228,805	**261.2**
MRes$_{\text{fixed}}$	6.44 s	62.92%	309,526	273.2
MRes$_{\text{variable}}$	**2.73 s**	58.43%	**160,032**	274.6

TABLE III: Planning statistics using the level-based expansion scheme. Maximum replanning times, number of expansions and trajectory costs are averaged over the same tasks as in Tab. II.

	Mean Maximum Replanning Time	Time > 1 s	Expansions	Costs
Uniform	0.65 s	16.85%	50,880	273.8
MRes$_{\text{fixed}}$	0.57 s	12.36%	26,078	**272.3**
MRes$_{\text{variable}}$	**0.30 s**	**6.74%**	**17,772**	276.2

To further investigate this behavior, we have a closer look at the tasks where uniform planning outperforms multiresolution planning. We compare the f-values of each state s, i.e., the sum of the costs for moving from the start state to s and the heuristic costs for reaching the goal from s. Fig. 6 shows histograms for the f-values of each expanded state during an example search for the different state lattice representations. The number of states grows approximately cubic with increasing f-value. Due to the reduced number of states, the curves for the multiresolutional lattices grow at lower rates. The A* algorithm expands all states whose f-value are smaller than the optimal path costs. Since the costs of the paths in multiresolutional lattices are higher than the optimal costs in the uniform lattice, states up to a higher f-value have to be expanded. Due to the cubic increase, this overhead might result in an overall larger amount of expanded states for multiresolutional search. Note that the difference between the maximum f-values for the different state representations is only 8. This corresponds to the smallest possible cost for one action. When reducing the time cost factor ρ, multiresolutional planning expands fewer states than uniform planning (Fig. 6 b), but the resulting trajectory durations might be suboptimal. Instead, the issue can be addressed by removing the constraint that all states whose f-value are smaller than the optimal path costs have to be expanded. This is evaluated in the next section.

C. Level-Based Expansion Scheme

We apply the level-based expansion scheme from Sec. III-D to all three state lattice representations and evaluate them on the same set of tasks that was used in Sec. IV-B. The results are summarized in Tab. III. All three methods benefit from using the adapted expansion scheme. While the average maximum replanning times for uniform state lattices are reduced by around 80%, the reduction for multiresolutional lattices is even higher with around 90%. Similar reductions are achieved with respect to the number of state expansions. The trajectory costs for uniform lattices increases by roughly 5%. Interestingly, the costs for multiresolutional planning do

TABLE IV: Comparison of the baseline [7] against the methods presented in this paper. Success denotes the fraction of tasks for which a valid solution was found within at most three million expansions. The number of tasks for which the longest replanning step exceeds 1 s includes tasks for which the corresponding method did not find a valid solution although one exists. Maximum replanning times, number of expansions and trajectory costs are averaged over the tasks where all methods found a valid solution. To compare the last three methods (using Level-A* and the 1D heuristic) against each other, please refer to Tab. III, which also considers more challenging tasks not solved by the baseline.

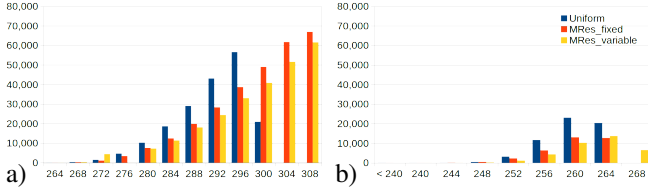|	Success	Mean Maximum Replanning Time	Time > 1 s	Expansions	Costs
Uniform, h_{base}, A*	71.43%	22.52 s	89.80%	1,085,740	**241.31**
MRes$_{fixed}$, h_{base}, A*	79.59%	17.87 s	89.80%	744,640	249.91
MRes$_{variable}$, h_{base}, A*	87.76%	14.21 s	87.76%	664,036	253.85
Uniform, h_{1D}, A*	91.84%	1.14 s	51.02%	70,085	243.16
MRes$_{fixed}$, h_{1D}, A*	95.92%	2.87 s	66.33%	143,822	254.33
MRes$_{variable}$, h_{1D}, A*	98.98%	1.60 s	62.24%	89,085	255.85
Uniform, h_{base}, Level-A*	95.92%	0.41 s	29.59%	24,190	246.15
MRes$_{fixed}$, h_{base}, Level-A*	97.96%	0.20 s	20.41%	**5684**	246.69
MRes$_{variable}$, h_{base}, Level-A*	**100%**	**0.19 s**	16.33%	8505	248.84
Uniform, h_{1D}, Level-A*	**100%**	0.20 s	22.45%	16,739	253.94
MRes$_{fixed}$, h_{1D}, Level-A*	**100%**	0.21 s	20.41%	8284	252.15
MRes$_{variable}$, h_{1D}, Level-A*	**100%**	0.20 s	**12.24%**	10,945	255.28

Fig. 6: Histograms for the f-values of each expanded state with different time cost weights ρ. a) $\rho = 16$. b) $\rho = 8$.
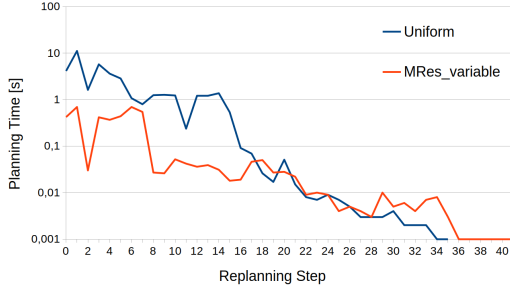

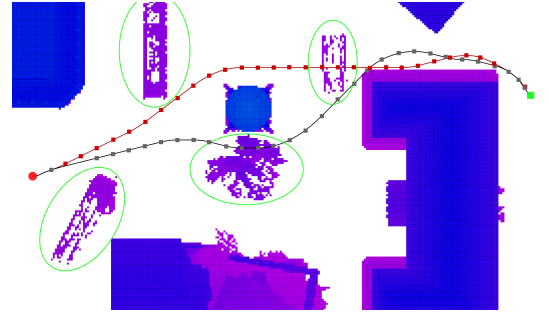Fig. 7: Replanning times in simulation.

Fig. 8: Trajectories from online replanning (top-down view). Obstacle heights are represented by the color, reaching from purple (low) to blue (high). The trajectory of MRes$_{variable}$ is shown in red, the trajectory from uniform planning in black. The start position is marked by the red circle and the goal by a green square. Start positions of replanning steps are marked with red and gray squares. The initially unknown obstacles are circled.

not change much. When using fixed level-dependent time steps, the costs even decrease slightly, which is possible because the used heuristic is not consistent.

In Tab. IV, we compare the method of [7] against all methods presented in this work. Note that average planning times, expansion numbers and trajectory costs only consider tasks solved by the baseline. The costs of all trajectories are at most 6% longer than the optimal trajectory generated by the baseline. However, the average maximum replanning time can be reduced by up to two orders of magnitude.

D. Online Replanning

Finally, we test our approach in simulation using the RotorS simulator [20]. We start with an initial OctoMap [21] of the environment (Fig. 4), but add additional unmapped static obstacles such that frequent replanning is necessary. Replanning is triggered at 1 Hz and the map is constantly updated using measurements of a simulated 3D laser scanner. Fig. 7 shows the corresponding planning times of uniform lattices and MRes$_{variable}$. Both methods use our 1D heuristic and the level-based expansion scheme. While our approach

has a maximal replanning time of 0.69 s, uniform planning exceeds 1 s in one third of all replanning steps. Fig. 8 shows the corresponding trajectories. A 3D view of the environment and the trajectory generated by our method is given in Fig. 1.

Additionally, we successfully employed our approach to online trajectory planning on a real MAV. For details on the integrated system and corresponding experiments, we refer to [22].

V. CONCLUSION

In this paper, we introduced high-dimensional local multiresolution state lattices. MAV velocities are directly incorporated into the planning to generate dynamically feasible trajectories. We showed that multiresolution combined with standard A* search might result in higher planning times compared to uniform state lattices. However, in combination with a level-based expansion scheme, multiresolutional state lattices significantly reduce the maximal planning times, while only moderately increasing trajectory costs. For some challenging tasks, planning times still exceed 1 s, but our approach was able to maintain a replanning frequency of 1 Hz for most cases. In summary, we demonstrated how multiresolution can increase the applicability of search-based high-dimensional trajectory planning for large dynamic environments.

REFERENCES

[1] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[2] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," *Computer Science Dept., Iowa State University*, 1998.

[3] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," in *Robotics Research*. Springer, 2016, pp. 649–666.

[4] E. Koyuncu and G. Inalhan, "A probabilistic B-spline motion planning algorithm for unmanned helicopters flying in dense 3D environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2008, pp. 815–821.

[5] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "STOMP: Stochastic trajectory optimization for motion planning," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 4569–4574.

[6] M. Nieuwenhuisen and S. Behnke, "Search-based 3D planning and trajectory optimization for safe micro aerial vehicle flight under sensor visibility constraints," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2019, pp. 9123–9129.

[7] S. Liu, N. Atanasov, K. Mohta, and V. Kumar, "Search-based motion planning for quadrotors using linear quadratic minimum time control," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 2872–2879.

[8] S. Liu, K. Mohta, N. Atanasov, and V. Kumar, "Search-based motion planning for aggressive flight in SE(3)," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2439–2446, 2018.

[9] S. Behnke, "Local multiresolution path planning," in *Robot Soccer World Cup*. Springer, 2003, pp. 332–343.

[10] M. Nieuwenhuisen and S. Behnke, "Hierarchical planning with 3D local multiresolution obstacle avoidance for micro aerial vehicles," in *Joint 45th International Symposium on Robotics (ISR) and 8th German Conference on Robotics (ROBOTIK)*. VDE, 2014.

[11] ——, "Local multiresolution trajectory optimization for micro aerial vehicles employing continuous curvature transitions," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 3219–3224.

[12] W. Du, F. Islam, and M. Likhachev, "Multi-Resolution A*," in *International Symposium on Combinatorial Search (SoCS)*, 2020.

[13] A. González-Sieira, M. Mucientes, and A. Bugarín, "An adaptive multi-resolution state lattice approach for motion planning with uncertainty," in *Robot 2015: Second Iberian Robotics Conference*. Springer, 2016, pp. 257–268.

[14] ——, "Graduated fidelity lattices for motion planning under uncertainty," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2019, pp. 5908–5914.

[15] M. Pivtoraiko and A. Kelly, "Differentially constrained motion re-planning using state lattices with graduated fidelity," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2008, pp. 2611–2616.

[16] O. Andersson, O. Ljungqvist, M. Tiger, D. Axehill, and F. Heintz, "Receding-horizon lattice-based motion planning with dynamic obstacle avoidance," in *IEEE Conference on Decision and Control (CDC)*, 2018, pp. 4467–4474.

[17] M. Likhachev and D. Ferguson, "Planning long dynamically feasible maneuvers for autonomous vehicles," *The International Journal of Robotics Research*, vol. 28, no. 8, pp. 933–945, 2009.

[18] M. Rufli, D. Ferguson, and R. Siegwart, "Smooth path planning in constrained environments," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2009, pp. 3780–3785.

[19] J. Petereit, T. Emter, and C. W. Frey, "Mobile robot motion planning in multi-resolution lattices with hybrid dimensionality," *IFAC Proceedings Volumes*, vol. 46, no. 10, pp. 158–163, 2013.

[20] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, "RotorS—A modular Gazebo MAV simulator framework," in *Robot Operating System (ROS)*. Springer, 2016, pp. 595–625.

[21] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, 2013. [Online]. Available: http://octomap.github.com

[22] D. Schleich, M. Beul, J. Quenzel, and S. Behnke, "Autonomous flight in unknown GNSS-denied environments for disaster examination," *arXiv preprint arXiv:2103.11742*, 2021.