SplatPlanner: Efficient Autonomous Exploration via Permutohedral Frontier Filtering

Anthony Brunel, Amine Bourki, Cédric Demonceaux, Olivier Strauss

SplatPlanner: Efficient Autonomous Exploration via Permutohedral Frontier Filtering

Anthony Brunel[1,2], Amine Bourki[2], Cédric Demonceaux[3] and Olivier Strauss[1]

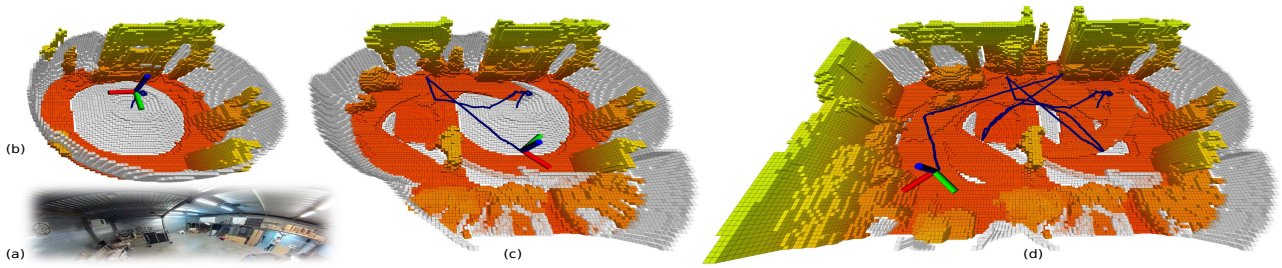


Fig. 1: We introduce *SplatPlanner*, a fast algorithm for jointly mapping and planning a collision-free exploration of an unknown scene, using an MAV. Here, our method runs *online* and *on-board* in **real-flight** (a) to gradually enrich a volumetric 3D map of the scene at different iterations, at $t = 54s$ (b), $t = 169s$ (c) and $t = 347s$ (d), near completion. Mapped voxels are colored based on their height, while *frontier voxels* that bound the known area are in gray, and trajectories in dark blue.

***Abstract*— We address the problem of autonomous exploration of unknown environments using a Micro Aerial Vehicle (MAV) equipped with an active depth sensor. As such, the task consists in mapping the gradually discovered environment while planning the envisioned trajectories in real-time, using on-board computation only. To do so, we present *SplatPlanner*, an end-to-end autonomous planner that is based on a novel Permutohedral Frontier Filtering (PFF) which relies on a combination of highly efficient operations stemming from bilateral filtering using permutohedral lattices to guide the entire exploration. In particular, our PFF is computationally linear in input size, nearly parameter-free, and aggregates spatial information about frontier-neighborhoods into density scores in one single step. Comparative experiments made on simulated environments of increasing complexity show our method consistently outperforms recent state-of-the-art methods in terms of computational efficiency, exploration speed and qualitative coverage of scenes. Finally, we also display the practical capabilities of our end-to-end system in a challenging real-flight scenario.**

***Index Terms*—Aerial Systems: Perception and Autonomy, Vision-Based Navigation, Motion and Path Planning**

I. Introduction

The safe exploration of unknown environments by unmanned vehicles is a key requirement to transition into the long-awaited era of autonomous flying robots. Micro Aerial Vehicles (MAVs) in particular are often favored thanks to their agility, form-factor and affordability. However, despite their perpetually improving hardware capabilities, MAVs still run on limited resources in terms of on-board computation and battery-life. This has motivated two orthogonal research trends to integrate these constraints through *collaborative exploration* [1], [2], and *efficiency-oriented* mono-agent methods [3], [4], which falls within our scope.

In this work, we devise a hybrid strategy relying on the two leading frameworks in autonomous exploration planning, frontier reasoning [5] and Rapidly-exploring Random Trees (RRT*s) [6], [7], [8], and advance the current state-of-the-art in terms of speed and efficiency by a substantial margin. **Our work makes the following main contributions:**

- We introduce *SplatPlanner*, an end-to-end system for autonomous exploration that simultaneously maps the environment and plans collision-free trajectories therein.
- The method uses a novel *Permutohedral Frontier Filtering* to extract and score promising frontier voxels in terms of their spatial density using highly efficient bilateral filtering operations on permutohedral lattices [9].
- Our proposed system advances the state-of-the-art w.r.t standard criteria of exploration speed and efficiency, as shown through quantitative and qualitative experiments in controlled simulated environments.
- The practical robustness of our approach is tested in a challenging real-flight scenario, also showing superior performance w.r.t state-of-the-art planners in this regard.

II. Related Work

The task of autonomous exploration planning is a long-standing problem and was studied for over two decades [5], [6], [10], [11]. The adopted methods can be categorized into frontier-based, sampling-based and hybrid strategies.

The first trend builds on the concept of *frontiers*, which are interstitial voxels between free- and unknown spaces (Fig. 1). It requires to re-detect such entities before each path-planning step. In their seminal work, Yamauchi et al. [5] target the nearest available frontier to guide the exploration, eventually leading to a termination when all reachable frontiers are visited. In the same vain, Cieslewski et al. [12] prioritize frontiers that maintain the velocity of the MAV, hence limiting visits to positions in opposing directions.

[1]LIRMM, Univ. Montpellier, CNRS, 860 rue de St Priest, 34095 Montpellier, France. {anthony.brunel, olivier.strauss}@lirmm.fr

[2]Gambi-M R&D, Paris, France. amine.bourki(@)gmail.com

[3]VIBOT EMR CNRS 6000, ImViA, Université Bourgogne Franche-Comté (UBFC), Le Creusot, France. cedric.demonceaux(@)u-bourgogne.fr

Dornhege et al. [13] adhere to the broader Next-Best-View (NBV) paradigm [14], i.e., selecting views that maximize a specific utility function, considering the popular *unmapped volume* within the current view frustum while also explicitly grouping frontiers into cohesive entities. Even though these two processes have become staple ingredients in frontier-based techniques [10], they respectively rely on ray-casting and spatial clustering operations that are computationally expensive. Other common utility functions include the information gain [15], Shannon entropy [16], [17], supervised learning [18], and many other efficient alternatives [19].

Sampling-based strategies on the other hand, randomly *sample* seed positions in space then maintain a data structure that can further be queried to produce a collision-free path connecting them. Most of the recent competitive methods rely on Rapidly-exploring Random Trees (RRTs) [6], [7]. The framework uses a tree-structure over the sampled seeds but can produce jagged trajectories. RRTs* [8] extend RRTs by biasing the random seed sampling to plan simpler, more efficient trajectories. RH-NBVP [20] expands RRTs on viewpoints that are sampled by means of NBV where the first node of the utility-maximizing branch is executed. While the method performs very well in local exploration, it is prone to local minima leading to a poor global scene coverage. Witting et al. [21] tackle this by maintaining a history of previously considered viewpoints to the expense of an additional cost in computation and storage. Alternatively, AEP [3] successfully combines RH-NBVP with conventional frontier reasoning to balance out the poor performance in global exploration. Although RRTs* are *effective* for collision-free planning, they cannot recover from earlier mistakes. They are also notoriously *inefficient* as they repeatedly discard-then-rebuild most of the information from one iteration to another. To improve efficiency, ESM [22] continuously grows one single tree and only sporadically queries feasible paths.

More related to our hybrid strategy, FFI [4] uniformly samples frontiers per octant in an octree structure [23], then targets views that maximize the map entropy. Despite achieving a very fast exploration, the strategy can suffer from quantization effects and induces a trade-off between time- and memory efficiency.

III. Problem Statement

Given $\mathcal{V} \subset \mathbb{R}^3$ a voxel-grid *occupancy map* of the scene, the goal of our considered MAV is to *map* every voxel $\mathbf{x} \in \mathcal{V}$ initially labeled as *'unknown'*, with a final volumetric occupancy among $\{$*'free', 'occupied'*$\}$ states, while computing collision-free flight-paths online and on-board. We denote the *pose* of the MAV by $\mu = [\boldsymbol{p}, \psi]$, where $\boldsymbol{p} = [x, y, z]^\top \in \mathbb{R}^3$ is a 3D position and $\psi \in [0, 2\pi[$ a corresponding yaw angle. We also assume a maximum linear velocity v_{max} and linear acceleration a_{max}, as well as a maximum yaw rate ω_{max}.

Our framework exploits the notion of *frontiers* [5] which are *'unknown'*-labeled voxels $\boldsymbol{f} \in \mathcal{F} \subset \mathcal{V}$ that are adjacent to *'free'* spaces. Our method extracts a set of filtered frontiers $\widetilde{\mathcal{F}} \subset \mathcal{F}$ using scores $s(\boldsymbol{p_i}) \subset \mathcal{S}$ per position $\boldsymbol{p_i} \in \mathcal{F}$ based on the density of their spatial neighborhoods. The MAV then repeatedly calculates collision-free paths online, across the scene until all reachable frontiers are visited.

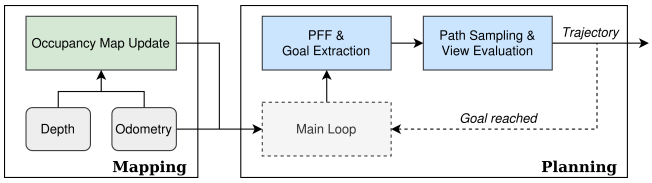


Fig. 2: **SplatPlanner –** System overview.

IV. SplatPlanner

We propose a hybrid framework that combines the advantages of frontier-reasoning (i.e., global scene coverage, natural termination) with the effectiveness of RRTs* in planning collision-free flight-paths. The number of costly calls to RRTs* is alleviated by an NBV-sampling based on our novel *Permutohedral Frontier Filtering (PFF)* procedure. Our method relies on two interconnected modules, as detailed below and illustrated in Fig. 2.

A mapping routine continuously integrates depth and odometry sensor information to update the occupancy map $\mathcal{V}$. The main loop of the planning unit activates the subsequent mechanisms: our PFF efficiently associates scores to every frontier $\boldsymbol{f} \in \mathcal{F}$ based on their spatial neighborhoods in $\mathcal{V}$. Next, the most promising goal candidate positions are extracted from the scored frontiers and are evaluated in terms of the *unknown*-labeled volume within corresponding frustums and the time to execute the corresponding trajectory. The output trajectory is then selected as the best trade-off between its associated UV and its execution time, translating into a natural balance between local and global exploration.

A. Mapping Routine

Occupancy Map and Derived Information. Map updates occur when new depth measurements are received from the active sensor (e.g., Kinect, or Intel Realsense D400 series). The occupancy values of voxels in $\mathcal{V}$ that lie inside the current view frustum are then directly inserted into $\mathcal{V}$ by using odometry data and assuming a standard probabilistic occupancy [24], [25]. While most of the mapping methods use ray-casting operations to maintain occupancy maps [23], [25], [26], [27], we chose the *projection mapping* which is an efficient alternative especially when dealing with regular grid maps. This is consistent with conclusions drawn in existing studies on the topic [28]. A simple Euclidean Distance Transform (EDT) applied to $\mathcal{V}$ [27], [29] is also used to derive a volumetric collision map to handle collision-checking. This allows efficient $O(1)$ look-ups vs. $O(log(N))$ for octrees [25]. Each update also leads to re-assessing frontiers $\mathcal{F}$ due to potential changes in *unknown*- and *free* voxels they separate.

Efficient Updates. While $\mathcal{V}$ and $\mathcal{F}$ need to be updated each time new depth information is received, the EDT map is continuously updated at a frequency of 4Hz, which is sufficient in practice to ensure its relevance when matching reasonably realistic MAV velocities. All of these updates are only envisioned within the current view-frustum of the MAV.

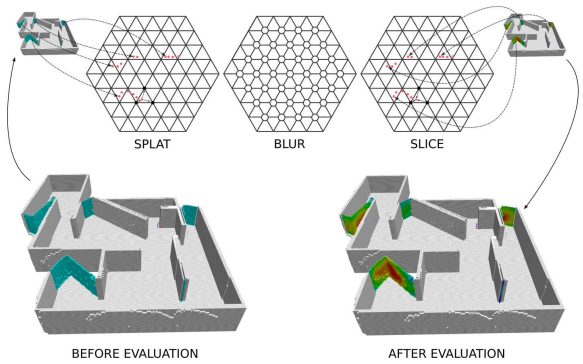


Fig. 3: **Permutohedral Frontier Filtering –** PFF transforms the frontier voxels through 3 successive bilateral filtering operations (*splat-blur-slice*) on permutohedral lattices, and produces their density scores efficiently.

B. Novel Permutohedral Frontier Filtering (PFF)

Overview and Input to PFF. The exploration planning process first starts by using our PFF to associate a score of frontier-neighborhood density to each frontier voxel. This score is the criterion we use to guide our planning. To do so, we use permutohedral lattice structures to support bilateral filtering operations to efficiently approximate this score. Permutohedral lattices are widely used in 2D image processing and geometry processing to do high dimensional filtering [9]. Other recent applications include, e.g., semantic segmentation of 3D point clouds using Deep Neural Networks [30], or fast point cloud registration [31]. A Permutohedral Lattice is composed of identical high-dimensional tetrahedra named *simplices*, where the enclosing simplex (i.e., cell) of any given point in the higher-dimensional space can be found by a simple rounding algorithm (as illustrated in Fig. 3). Such structures have been proven to yield very accurate approximations to expensive filtering problems for even high $d-$dimensional inputs by processing them in a $(d+1)-$dimensional space. It uses very efficient operations requiring only a fraction of the computation [32].

As such, our PFF consumes the 3D coordinates of the frontier set $\mathcal{F}$ and goes through three operations, *splat, blur* and then *slice*, each-of-which being supported by a lattice.

Processing Steps and Parameters. *Splat* first interpolates the input 3D coordinates onto their enclosing 4D permutohedral simplices using barycentric weights. *Blur* then applies a separable Gaussian filter resulting in a weighted combination of entities with their simplicial neighbors which equates to an implicit spatial aggregation of features. *Slice* finally interpolates the lattice features back onto their original 3D coordinates with corresponding densities, using the same weights used in splatting. The only parameter to the method is a scaling diagonal $d \times d$ matrix Λ that controls the discretization (spacing) of the simplicial cells (here, $d = 3$).

Formally, the sequence of operations can be written as a matrix multiplication, as follows:

$$\Delta = E_{slice} B_{blur} S_{splat} F, \tag{1}$$

where Δ contains per-frontier filtered density scores, F is the $N \times 3$ matrix version of frontiers $\mathcal{F}$ and is transformed through three $N \times 4$ matrices in higher dimensional space $E_{slice}, B_{blur}, S_{splat}$ respectively supporting *slice, blur, splat* operations, and $N = card(\mathcal{F})$.

Advantages of our PFF. The procedure has the following main benefits.

- *Splat, blur* and *slice* are linear in frontier inputs and their combination amounts to multiplying sparse matrices.
- Through the *Blur* operation in particular, the process seamlessly generates an implicit grouping, i.e., a neighborhood aggregation of frontiers in linear time. *Splat and Slice* are just means to interact with the initial 3D space and amount to a basic linear interpolation each. This avoids the costly spatial clustering used by several frontier-based planners altogether [13], [33].
- PFF is only parametrized by the scaling matrix Λ which is a diagonal matrix of inverse values of the desired standard deviations ($stds$) per spatial coordinate. Throughout our experiments, we consider one common std for all three coordinates as $\sigma_{xyz} = 1m$.
- The framework allows to handle much more complex features in dimensionality on top of the 3D coordinates we consider [30]. We leave this avenue as future work, e.g., to integrate semantics to our exploration formulation.

Additional details and advantages regarding the general permutohedral filtering framework can be found in the dedicated study of Baek et al. [32].

C. Goal Extraction and Path Sampling

The output of the preceding PFF scoring is used to extract a list of promising goal positions to visit. To do so, the *goal extraction* aims at favoring areas with high density scores evenly spread across the space to ultimately bound the computed feasible RRT*-paths from sparsely-sampled goals.

The goal extraction operation assumes positions within the frontier set, ranked in decreasing density scores and iterates over frontiers by only retaining points that are far enough from previously retained positions (up to a radius threshold r_{thresh}). The detailed algorithm is provided in Algo. 1. Finally, the set of retained goals is formed by considering a uniform sampling of $N_{uniform}$ extracted frontiers across the space, augmented by the top N_{scores} scored entities remaining after the aforementioned spatial filtering to prevent near-duplicates to occur. Our random selection criterion ensures an even distribution of goals and secures the inclination towards global exploration. The density-favoring criterion on the other hand, guarantees that the most promising areas are not left out, regardless of their spatial distribution. In our experience, this yields a much better trade-off in practice between local- and global exploration than pure uniform sampling [4] or pure local development [3].

Once the limited set of promising goals G is computed, we plan their feasible trajectories $P_i = [\boldsymbol{p}, ..., \boldsymbol{p}_i]$ stemming from the current MAV position using RRT* [8] with path simplification, as implemented in the Open Motion Planning Library (OMPL) [34]. A standard collision checking is also

Algorithm 1: Goal Candidate Extraction

Input: $\mathcal{P}$: list of position / score pairs $(\boldsymbol{p_i}, s_i)$ in decreasing scores, r_{thresh} : radius threshold
Output: G : set of goal candidates
foreach *position* $\boldsymbol{p_i} \in \mathcal{P}$ **do**
  $discarded \leftarrow False$;
  foreach $\boldsymbol{p_j} \in G$ **do**
    if $\|\boldsymbol{p_i} - \boldsymbol{p_j}\| < r_{thresh}$ **then**
      $discarded \leftarrow True$;
      $break$;
    end
  end
  if *not discarded* **then**
    $G \leftarrow (\boldsymbol{p_i}, s_i)$
  end
end

applied throughout the linear segments of the computed paths with a safety bounding radius R centered on the MAV. An occlusion checking by casting a ray from the current position to the goal candidates ensures mutual visibility between end-points. An additional time-out forces the RRT* to provide viable solutions within a limited time (set to 5 ms), for maintaining the overall pace of the exploration.

D. Viewpoint Evaluation, Yaw Angles and Gain Function

At this stage, the promising goals resulting from previous steps are guaranteed by construction to be valid and reachable. Next, we describe how we prioritize the exploration to maximize its *efficiency* by evaluating the volumetric gain relative to the time it costs to execute the associated trajectories. This consists in estimating through ray-casting the view-dependent *Unmapped Volume (UV)* intersecting the frustum bounded between distances to the camera center d_{min} and d_{max}. α (resp. β) is the horizontal (resp. vertical) angular increment used in ray-casting (typically, $\alpha = \beta = 1°$). Thus, a 360° horizontal UV-sweep produces a view-*independent* estimate of the volume assuming a fixed (sensor-specific) vertical fov, as follows:

$$V(d_{min}, d_{max}, \alpha, \beta) = 4 \tan(\frac{\alpha}{2}) \tan(\frac{\beta}{2}) \int_{d_{min}}^{d_{max}} r^2 dr$$
$$= \frac{4}{3} \tan(\frac{\alpha}{2}) \tan(\frac{\beta}{2}) (d_{max}^3 - d_{min}^3) \quad (2)$$

Yaw Optimization. Yaw values ψ_i are associated to the positions $\boldsymbol{p_i}$ encompassing goal candidates and intermediate waypoints alike, by retaining the horizontal orientation that maximizes its view-dependent UV within the computation of the volume. Please note that the optimal yaws are retained while calculating the volume with no additional post-process.
Gain Function. Finally, the gain function explicitly optimizes the *volumetric exploration speed* in m^3/s, to execute the path P_i that maximizes the following expression:

$$g(\mu_i, P_i) = \frac{\mathcal{U}(\mu_i)}{T(P_i)}, \quad (3)$$

TABLE I: Experimental settings and parameters.

Method	Parameter	MAZE	POWERPLANT	FACILITY
All	$res.\ (m)$	0.1	0.2	0.1
	$R\ (m)$	0.5	0.5	0.5
	$v_{max}\ (m/s)$	1.5	$0.7, 1.5, 2.5$	1.5
	$a_{max}\ (m/s^2)$	2.5	2.5	2.5
	$\omega_{max}\ (rad/s)$	1.57	$0.75, 1.57$	1.57
	$d_{max}\ (m)$	5	7	5
	$fov\ [h, v]$	$[90°, 60°]$	$[115°, 60°]$	$[90°, 75°]$
Ours	$r_{thresh}\ (m)$	2.5	2.5	2.5
	$\sigma_{xyz}\ (m)$	1.0	1.0	1.0
AEP [3]	N	30	50	30
	g_{zero}	2	2	2
	λ	0.5	0.75	0.5
	σ^2_{tresh}	0.2	0.2	0.2
	$l_{max}\ (m)$	1.5	3	1.5
ESM [22]	n_{local}	30	50	30
	$r_{local}\ (m)$	1.5	3.0	1.5
	$r_{update}\ (m)$	3.	6.	3.0
	$l_{max}\ (m)$	1.5	3	1.5

where $\mathcal{U}(\mu_i)$ is the UV of the destination pose at position $\boldsymbol{p_i}$ and P_i its connecting path to current MAV position $\boldsymbol{p}$ with pose μ. In turn, $T(P_i)$ is the total travel time to execute P_i assuming a constant acceleration a_{max}.

V. IMPLEMENTATION DETAILS

Simulation Framework. Except where stated otherwise, throughout our experiments we consider a common testbed built on the Robot Operating System (ROS Kinetic) library [35] on a laptop equipped with an Intel Core i7 CPU, 16GB RAM. We simulate a FireFly hexacopter MAV model from Ascending Technologies and its physics are modeled by the popular Gazebo-based RotorS simulator [36]. The control is insured by a multi-rotor Model Predictive Control (MPC) [37]. Standard collision-checking is done within a sphere of radius R centered at the MAV's position. As the computed trajectories are typically piecewise-linear displacements, we use a polynomial trajectory optimization [38] to send graceful paths to the MPC. The final rendering uses the Vulkan API[1], assuming perfect poses from RotorS. It simulates a 30Hz DepthVision camera. All considered datasets were integrated as standard meshes into Gazebo.
Parameter Settings and Reproducibility. The parameters to all our experiments are provided in Table I, including ones that were originally omitted by the referenced competition.

VI. RESULTS

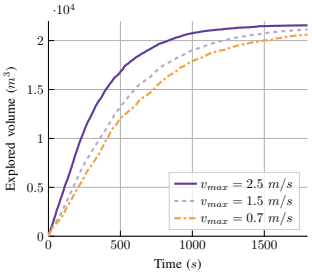
In this section, we analyze the performance of our SplatPlanner system and its relative positioning w.r.t the state-of-the-art through quantitative and qualitative experiments.

A. Datasets and Baselines

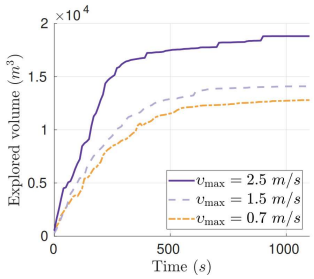
In our comparative experiments, we consider the recent state-of-the-art planners ESM [22], AEP [3] and FFI [4]. ESM is the current top-performing pure sampling-based method [22] while AEP and FFI are among the best hybrid systems for autonomous exploration.

We first consider 3 challenging synthetic datasets of increasing complexity and size.

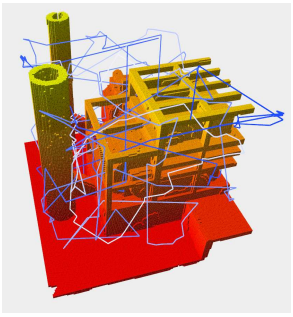
[1] https://www.khronos.org/vulkan/



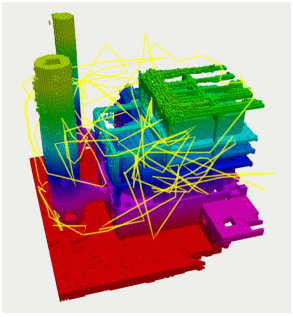
(a) SplatPlanner **(Ours).**



(b) FFI [4] (As captured from their paper).



(c) SplatPlanner **(Ours)** achieves a simpler trajectory and a better coverage.



(d) FFI [4] (As captured from their paper).

Fig. 4: Comparative evaluation vs. FFI [4] on the POWERPLANT simulation in terms of exploration speed (a–b), corresponding trajectories and produced maps (c–d). Our SplatPlanner achieves faster explorations throughout the considered maximum exploration velocities v_{max}.

Fig. 5: Exploration of the MAZE after $250s$ using our approach. Voxels are colored based on their height, while the trajectory ranges progressively from white to blue.

- MAZE – A hand-crafted scene of 20m x 20m x 2.5m typically used to study exploration systems by narrowing down their behaviors to their most basic expression (Fig. 5).
- POWERPLANT – A large outdoor of 33m x 31m x 26m. It is publicly available and was used by FFI [4] in particular. In absence of a public code of their method it is a common test-bed to compare our performances (Fig. 4c).
- COMPLEX FACILITY – An original challenging CAD model we are sharing publicly. It covers 20m x 20m x 4.6m amounting to a 185m^2 area and depicts a very geometrically complex industrial scene that has been modeled by a human expert (Fig. 6a).

Please note that in terms of datasets, our experimental setup is at least on par *quantity-wise* with the most rigorously evaluated works in the field (e.g., [12], [3]). Our COMPLEX FACILITY however, is significantly more complex and challenging than commonly envisioned datasets [3], [4], [20].

B. Comparative Setup and Evaluation Criteria

Autonomous exploration algorithms are typically evaluated on synthetic datasets in controlled simulated environments. This allows for a more fair comparison in terms of experimental conditions and also avoids the expensive failure cases in real flights. A detailed description of the simulation framework is provided in Section V. First, we analyze performance vs. ESM [22] and AEP [3] by integrating their respective public implementations[2] to our experimental setup to allow for a fair comparison, as previously discussed.

Next, in absence of a public implementation for FFI [4], we modify our experimental setup to match the assumptions they disclose in the corresponding paper, and propose a fair side-by-side comparison on the public POWERPLANT dataset using the authors' reported performances and illustrations (Fig. 4b, 4d). In particular, we replace our MPC by its RotorS-based counterpart and use the reported parameters (Fig. 4a, 4c, Table I).

Performance is evaluated w.r.t to the following criteria.

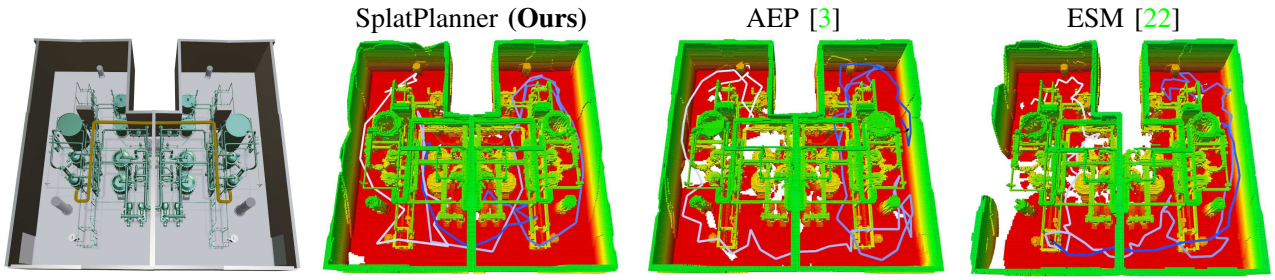
- **Exploration speed** is measured quantitatively by means of the explored volume over time, averaged over 10 runs to account for the stochastic nature of the methods. This is the commonly adopted practice in the field.
- **Exploration efficiency** assesses the ability of a system to *make the most out of its exploration*. *Quantitatively* this equates to the final (plateau) volumetric coverage in terms of mapped volume. *Qualitatively*, this is assessed visually by comparing the covered areas.

C. Positioning vs. State-of-the-art

Pure Performance. Our SplatPlanner significantly and consistently exhibits superior performance w.r.t the considered state-of-the-art methods ESM and AEP throughout the experiments in exploration speed (Fig. 7a, 7b, 6b). The tendency is also confirmed w.r.t the final explored volume (i.e., efficiency) except on the COMPLEX FACILITY where ESM seems on par with SplatPlanner. W.r.t FFI on the POWERPLANT, SplatPlanner also performs consistently better throughout different maximum velocities (Fig. 4a, 4b), except before the 500s mark where the competition is slightly faster for $v_{max} = 2.5\ m/s$. Beyond this limit, the data favors the proposed approach leading to a far superior efficiency (final explored volume). This tendency is confirmed by the qualitative scene coverage as well (Fig. 4c, 4d).

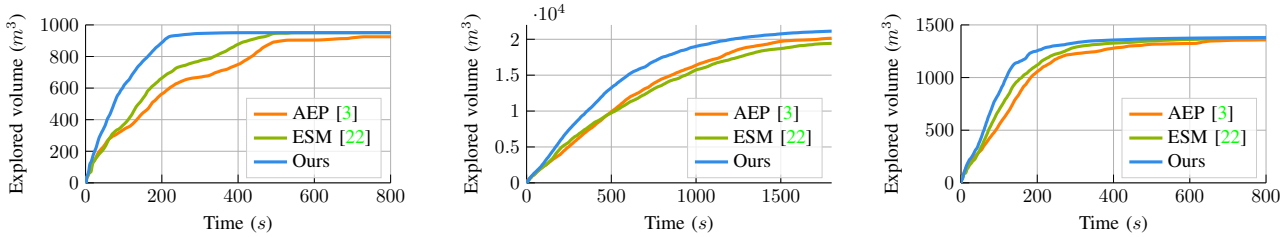
Parameter Settings and Robustness. Table I discloses all the parameters used in our experiments, including parameters from the competition that were unspecified in their respective papers. SplatPlanner only requires 2 trivial hyper-parameters

[2] https://github.com/ethz-asl/mav_active_3d_planning and https://github.com/mseln/aeplanner



(a) CAD Input. (b) Qualitative comparison of the maps produced at $t = 300s$. Voxels are colored based on their height.

Fig. 6: COMPLEX FACILITY – Our SplatPlanner explores this complex environment more efficiently and better retrieves complex geometric details w.r.t the state-of-the-art competition.



(a) MAZE (b) POWERPLANT (c) COMPLEX FACILITY

Fig. 7: Exploration speeds averaged over 10 runs.

(vs. 5–6) that remain stable on all scenes. This supports the superior robustness of SplatPlanner w.r.t state-of-the-art in this aspect as well.

Additional Insights. Table II shows comparative exploration planning statistics w.r.t state-of-the-art. Our method requires much fewer, but slower planning iterations. Yet, it develops faster, into a much longer path overall.

TABLE II: COMPLEX FACILITY – Statistics after $300s$.

Method	#Iterations	Per-iter. avg time (ms)	Path length (m)
Ours	48	280 ± 34	**157**
AEP [3]	90	54 ± 25	124
ESM [22]	128	N/A	128

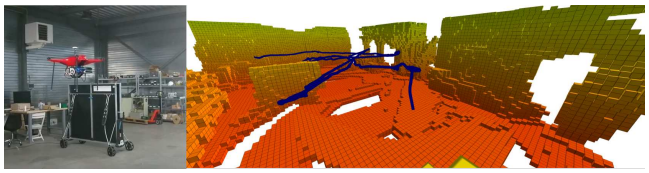


Fig. 8: *SplatPlanner* runs online and on-board from take-off to termination with no human intervention nor prior knowledge about this large 15m x 10m x 3.5m WAREHOUSE.

D. Real-Flight Performance

We evaluate the practical robustness of our end-to-end system on a challenging *fully-autonomous flight* (Fig. 8). Our custom MAV flies through a large WAREHOUSE that is bound at 15m x 10m x 3.5m and runs our SplatPlanner algorithm online and on-board during $349s$. The results of this experiment are depicted step by step in Fig. 1 and are the *direct output* of on-board processing. In comparison to real-flight experiments provided by top-performing planners [3], [4], ours takes place in a significantly more complex, larger indoor environment than the ones referenced. This represents an improvement in volumetric size of the considered scenes ranging from 2.75x [4] to 4.66x [3] over the leading competition. In terms of hardware, the MAV we consider is a custom quadrotor using a DJI A3 flight controller. The equipped sensors include an Intel T265 VI-SLAM for odometry, a D455 depth VGA camera and an Intel NUC with an Intel Core i7-10710U and 16GB of memory.

Additional detailed information about the real-flight and simulation experiments are provided in a supplementary video at https://youtu.be/DCcfA2HB1GI.

VII. CONCLUSION

We have presented *SplatPlanner*, an end-to-end autonomous system for 3D exploration planning in unknown environments using an MAV equipped with a depth sensor. The proposed method improves over recent state-of-the-art methods in terms of exploration efficiency and speed, as consistently supported by the considered simulation experiments. An additional real-flight scenario also underlines the practical robustness of our global solution.

As future work, we plan on integrating semantic reasoning into our formulation by leveraging the pose and volumetric occupancy of detected objects of interest.

REFERENCES

[1] B. Yamauchi, "Frontier-based exploration using multiple robots," in *Proceedings of the second international conference on Autonomous agents*, 1998, pp. 47–53.

[2] S. Dong, K. Xu, Q. Zhou, A. Tagliasacchi, S. Xin, M. Nießner, and B. Chen, "Multi-robot collaborative dense scene reconstruction," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 4, pp. 1–16, 2019.

[3] M. Selin, M. Tiger, D. Duberg, F. Heintz, and P. Jensfelt, "Efficient autonomous exploration planning of large-scale 3-d environments," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1699–1706, 2019.

[4] A. Dai, S. Papatheodorou, N. Funk, D. Tzoumanikas, and S. Leutenegger, "Fast frontier-based information-driven autonomous exploration with an mav," 2020.

[5] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97.'Towards New Computational Principles for Robotics and Automation'*. IEEE, 1997, pp. 146–151.

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

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

[8] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[9] A. Adams, J. Baek, and M. A. Davis, "Fast high-dimensional filtering using the permutohedral lattice," in *Computer Graphics Forum*, vol. 29, no. 2. Wiley Online Library, 2010, pp. 753–762.

[10] D. Holz, N. Basilico, F. Amigoni, and S. Behnke, "Evaluating the efficiency of frontier-based exploration strategies," in *ISR 2010 (41st International Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics)*. VDE, 2010, pp. 1–8.

[11] M. Juliá, A. Gil, and O. Reinoso, "A comparison of path planning strategies for autonomous exploration and mapping of unknown environments," *Autonomous Robots*, vol. 33, no. 4, pp. 427–444, 2012.

[12] T. Cieslewski, E. Kaufmann, and D. Scaramuzza, "Rapid exploration with multi-rotors: A frontier selection method for high speed flight," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 2135–2142.

[13] C. Dornhege and A. Kleiner, "A frontier-void-based approach for autonomous exploration in 3d," in *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*.

[14] C. Connolly, "The determination of next best views," in *Proceedings. 1985 IEEE international conference on robotics and automation*, vol. 2. IEEE, 1985, pp. 432–435.

[15] L. Heng, A. Gotovos, A. Krause, and M. Pollefeys, "Efficient visual exploration and coverage with a micro aerial vehicle in unknown environments," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 1071–1078.

[16] E. Palazzolo and C. Stachniss, "Information-driven autonomous exploration for a vision-based mav," *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 4, p. 59, 2017.

[17] ——, "Effective exploration for mavs based on the expected information gain," *Drones*, vol. 2, no. 1, p. 9, 2018.

[18] B. Hepp, D. Dey, S. N. Sinha, A. Kapoor, N. Joshi, and O. Hilliges, "Learn-to-score: Efficient 3d scene exploration by predicting view utility," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 437–452.

[19] F. Bissmarck, M. Svensson, and G. Tolt, "Efficient algorithms for next best view evaluation," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 5876–5883.

[20] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart, "Receding horizon" next-best-view" planner for 3d exploration," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 1462–1468.

[21] C. Witting, M. Fehr, R. Bähnemann, H. Oleynikova, and R. Siegwart, "History-aware autonomous exploration in confined environments using mavs," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1–9.

[22] L. Schmid, M. Pantic, R. Khanna, L. Ott, R. Siegwart, and J. Nieto, "An efficient sampling-based method for online informative path planning in unknown environments," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1500–1507, 2020.

[23] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous robots*, vol. 34, no. 3, pp. 189–206, 2013.

[24] C. Loop, Q. Cai, S. Orts-Escolano, and P. A. Chou, "A closed-form bayesian fusion equation using occupancy probabilities," in *2016 Fourth International Conference on 3D Vision (3DV)*. IEEE, 2016, pp. 380–388.

[25] E. Vespa, N. Nikolov, M. Grimm, L. Nardi, P. H. Kelly, and S. Leutenegger, "Efficient octree-based volumetric slam supporting signed-distance and occupancy mapping," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1144–1151, 2018.

[26] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning," in *2017 Ieee/rsj International Conference on Intelligent Robots and Systems (iros)*. IEEE, 2017, pp. 1366–1373.

[27] V. Usenko, L. von Stumberg, A. Pangercic, and D. Cremers, "Real-time trajectory replanning for mavs using uniform b-splines and a 3d circular buffer," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 215–222.

[28] M. Klingensmith, I. Dryanovski, S. S. Srinivasa, and J. Xiao, "Chisel: Real time large scale 3d reconstruction onboard a mobile device using spatially hashed signed distance fields." in *Robotics: science and systems*, vol. 4. Citeseer, 2015, p. 1.

[29] P. F. Felzenszwalb and D. P. Huttenlocher, "Distance transforms of sampled functions," *Theory of computing*, vol. 8, no. 1, pp. 415–428, 2012.

[30] H. Su, V. Jampani, D. Sun, S. Maji, E. Kalogerakis, M.-H. Yang, and J. Kautz, "Splatnet: Sparse lattice networks for point cloud processing," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2530–2539.

[31] W. Gao and R. Tedrake, "Filterreg: Robust and efficient probabilistic point-set registration using gaussian filter and twist parameterization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11 095–11 104.

[32] J. Baek and A. Adams, "Some useful properties of the permutohedral lattice for gaussian filtering," *other words*, vol. 10, no. 1, p. 0, 2009.

[33] C. Dornhege and A. Kleiner, "A frontier-void-based approach for autonomous exploration in 3d," *Advanced Robotics*, vol. 27, no. 6, pp. 459–468, 2013.

[34] I. A. Sucan, M. Moll, and L. E. Kavraki, "The open motion planning library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, 2012.

[35] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[36] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, *Robot Operating System (ROS): The Complete Reference (Volume 1)*. Cham: Springer International Publishing, 2016, ch. RotorS—A Modular Gazebo MAV Simulator Framework, pp. 595–625.

[37] M. Kamel, T. Stastny, K. Alexis, and R. Siegwart, "Model predictive control for trajectory tracking of unmanned aerial vehicles using robot operating system," in *Robot Operating System (ROS) The Complete Reference, Volume 2*, A. Koubaa, Ed. Springer.

[38] 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.

[39] H. Oleynikova, M. Burri, Z. Taylor, J. Nieto, R. Siegwart, and E. Galceran, "Continuous-time trajectory optimization for online uav replanning," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 5332–5339.

[40] B. Zhou, F. Gao, L. Wang, C. Liu, and S. Shen, "Robust and efficient quadrotor trajectory generation for fast autonomous flight," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3529–3536, 2019.

[41] C. Papachristos, S. Khattak, and K. Alexis, "Uncertainty-aware receding horizon exploration and mapping using aerial robots," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 4568–4575.

[42] H. H. González-Banos and J.-C. Latombe, "Navigation strategies for

exploring indoor environments," *The International Journal of Robotics Research*, vol. 21, no. 10-11, pp. 829–848, 2002.

[43] B. Adler, J. Xiao, and J. Zhang, "Autonomous exploration of urban environments using unmanned aerial vehicles," *Journal of Field Robotics*, vol. 31, no. 6, pp. 912–939, 2014.

[44] A. C. Schultz, W. Adams, and B. Yamauchi, "Integrating exploration, localization, navigation and planning with a common representation," *Autonomous Robots*, vol. 6, no. 3, pp. 293–308, 1999.

[45] T. Duckett and U. Nehmzow, "Exploration of unknown environments using a compass, topological map and neural network," in *Proceedings 1999 IEEE International Symposium on Computational Intelligence in Robotics and Automation. CIRA'99*. IEEE, 1999, pp. 312–317.

[46] B. Charrow, G. Kahn, S. Patil, S. Liu, K. Goldberg, P. Abbeel, N. Michael, and V. Kumar, "Information-theoretic planning with trajectory optimization for dense 3d mapping." in *Robotics: Science and Systems*, vol. 11, 2015.

[47] S. Song and S. Jo, "Surface-based exploration for autonomous 3d modeling," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–8.

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

[49] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 2520–2525.

[50] K. Xu, L. Zheng, Z. Yan, G. Yan, E. Zhang, M. Niessner, O. Deussen, D. Cohen-Or, and H. Huang, "Autonomous reconstruction of unknown indoor scenes guided by time-varying tensor fields," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 6, pp. 1–15, 2017.