MonoSpheres: Large-Scale Monocular SLAM-Based UAV Exploration through Perception-Coupled Mapping and Planning

Anonymous Authors

***Abstract*— Autonomous exploration of unknown environments is a key capability for mobile robots, but it is largely unsolved for robots equipped with a single monocular camera and no dense range sensors. In this paper, we present a novel approach to monocular vision-based exploration that can safely cover large-scale unstructured indoor and outdoor environments by explicitly accounting for the properties of a sparse monocular SLAM frontend in both mapping and planning. The mapping module solves the problems of sparse depth data, free-space gaps, and large depth uncertainty by oversampling free space in texture-sparse areas and keeping track of obstacle position uncertainty. The planning module handles the added free-space uncertainty through rapid replanning and forced forward flight. We further show that frontier-based exploration is possible with sparse monocular depth data when parallax requirements and the possibility of textureless areas are taken into account. We evaluate our approach extensively in various real-world and simulated environments and ablation studies. To the best of the authors' knowledge, the proposed method is the first to achieve 3D monocular exploration in real-world unstructured outdoor environments. We open-source the code of the proposed method to benefit the community.**

Code— In attached multimedia materials

I. INTRODUCTION

In recent years, significant advances have been presented in mobile robot autonomy, enabling robot systems to explore unknown 3D environments that span kilometers in scale [1]–[3]. These advances are critical for allowing the deployment of uncrewed aerial vehicles (UAVs) in real-world applications, such as search-and-rescue, wildfire response or planetary exploration. However, the mapping and exploration methods used for autonomy in unknown environments currently rely on dense range sensors, such as LiDARs or depth cameras. Such sensors are heavy, expensive, and thus severely limit the affordability and flight time of autonomous UAVs. Monocular cameras and inertial measurement units (IMUs), on the other hand, are orders of magnitude lighter and less expensive.

Monocular exploration of unstructured 3D environments has thus far been presented only at the scale of a few indoor rooms [4]–[7]. The existing monocular mapping approaches [4], [5] use the same core approach which is used with dense range sensors: [NO! implicit!] They build a grid-based map where cells are set as free if they are passed by rays that go from the sensor towards points that likely lie on some physical surface. Some methods [4], [5] trace rays towards sparse pointclouds generated by monocular SLAM, which leads to large gaps in free space, prohibiting planning in texture-sparse areas. Other methods [6], [8] use depth neural networks to obtain denser pointclouds. As confirmed by the authors, however, the deep neural depth estimates currently lack the reliablity for safe deployment and can often lead to the UAV crashing.

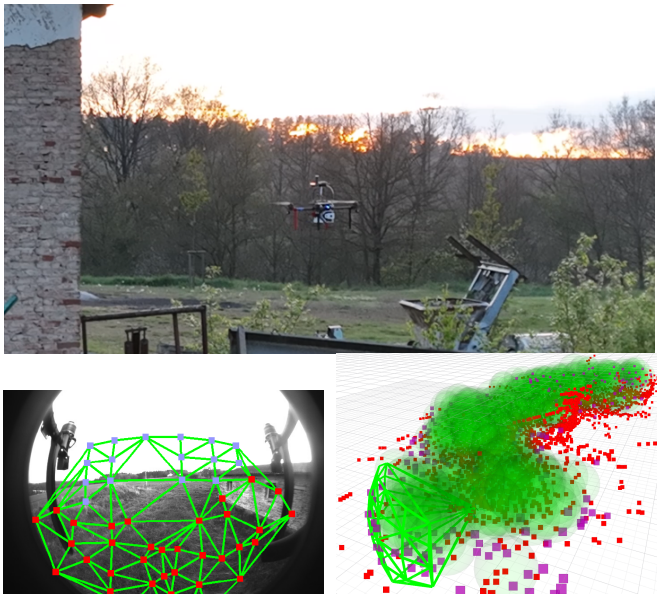

Fig. 1: Illustration of the proposed approach. The mapping pipeline estimates depth using virtual (blue) and tracked (red) points from sparse monocular-inertial SLAM (left) running onboard the UAV (top) to construct a large-scale map consisting of free-space spheres and obstacle points (right)

The core idea of our paper is that if the properties of sparse monocular SLAM are explicitly taken into account in designing both the mapping and the planning parts of an exploration system, its performance can be superior to methods that apply dense mapping approaches to the sparse depth frontend. TODO - shit on neural nets as well. Intersecting spheres are connected together to form a graph-of-spheres map representation that has been shown to allow rapid and safety-aware planning [9]–[11]. This map is built online, directly from the sparse pointclouds, without requiring any grid-based map to be built first.

Combined with the sampling scheme described in subsection III-B, this approach can accurately estimate free space even in low-texture areas where the traditional raytracing approach might not initialize sufficient free space to allow flight, as shown in our experimental results in subsection V-B. By storing obstacles as a set of geometric primitives, we can also assign additional metrics to each obstacle, such as measurement distance. This allows for rule-based measurement fusion, which is critical for dealing with problems such

as occlusions in monocular depth estimation, as we explain in subsection III-C. In several experiments, we show how this mapping approach, combined with perception-aware planning, enables exploration of large-scale unstructured areas which feature both textureless wide open spaces and cluttered passages.

As a step towards low-cost vision-based UAVs safely and efficiently navigating any large-scale, unknown 3D environments, our paper brings the following contributions:

1) A novel conceptual approach to 3D mapping that constructs a sphere-based map for safety-aware navigation using only sparse pointclouds and pose estimates from monocular-inertial SLAM as input, with the ability to map sufficient free space for flight even in texture-sparse open-space areas.
2) A perception-aware exploration approach, enabled by the proposed mapping method, that achieves large-scale 3D outdoor exploration on a UAV equipped with only a monocular camera and IMU, validated in real-world conditions.
3) Open-sourced code for the proposed methods, along with example simulation scripts for replicating our results, providing a benchmark for new monocular exploration methods.

II. Related Works

Exploration of unknown environments is a well-studied research problem [**?**], [**?**], [**?**] in the case of robots with dense range sensors, such as LiDARs or depth cameras. These works implicitly assume that the distance measurements from the sensor are dense and accurate enough to build a reliable occupancy map by tracing rays from the sensor origin towards the measured points and classifying the traversed space as occupied or free [12]–[15]. In essence – there is true occupied and free space, and the sensor "uncovers" all visible occupied and free space in the sensor's field-of-view and range at any given pose.

However, this assumption is not applicable when using a monocular camera, as there is currently no method that can reliably provide dense and accurate depth measurements in all real-world environments. [TODO-monocular depth is still an open problem, largest issue is GENERALIZATION - cite surveys.] [TODO-depth from motion using SLAM requires TEXTURE, and is sparse even when dense. + this adds a TRANSLATION to map space, while dense sensors dont require motion to get depth.]

Without this assumption, the problem of mapping and planning becomes significantly more challenging – undetected obstacles can lead to collisions, while undetected free space can lead to overly conservative plans that prevent exploration. Thus, it is not possible to directly apply the same mapping and planning approaches as with dense range sensors and the existing exploration approaches either fail to explore large-scale environments or make limiting assumptions on the environment structure. Existing approaches to solving this problem can be broadly divided into three main categories: methods that use monocular SLAM for sparse depth estimation, methods using monocular depth neural networks to turn the monocular camera into a dense depth sensor, or implicit methods that do not build an explicit 3D occupancy map.

navigation vs exploration, also STRUCTURE-ASSUMPTION methods

The authors of [6], [7] have demonstrated using such depth estimation models combined with traditional navigation approaches. However, the authors of [6] demonstrate autonomy only at the scale of a single room, while also encountering collisions and map deformations due to depth estimation errors. The authors of [8] present navigation with obstacle avoidance using pretrained and finetuned monocular depth models in larger outdoor environments in simulation, but also note a high collision and navigation failure rate due to the same problems. As further discussed in recent surveys [16], [17], current depth estimation models still struggle to provide real-time onboard performance and reliable generalization in domains that the model was not trained on. In addition, they require a GPU which significantly raises the cost and weight of a UAV.

A different line of research focuses on taking sparse 3D points estimated by visual SLAM and working with those as the only depth measurements. Most of this research has been on reactive obstacle avoidance [18], [19] or exploration with strong assumptions on the environment structure (e.g. corridors only [20] or considering a single object with no other obstacles [21]). For general monocular exploration in unstructured environments, the authors of [4], [5] propose building grid-based maps by tracing rays of free space towards the monocular SLAM points. As documented in [4], this approach causes gaps to appear in the free space of the map when the environment is not texture-rich. The authors of [4] further present an exploration method specifically to cover these gaps and demonstrate an increase in mapped volume. However, the authors show this exploration approach to only work at the scale of a single room. Most importantly though, this mapping approach fundamentally cannot estimate sufficient free space for navigation in texture-sparse areas, especially outdoor (as shown in subsection V-B). The authors of [5] also trace rays towards sparse SLAM points, and they present exploration on the scale of multiple indoor rooms in simulation. They work around the gap issue by setting the map voxel size to be relatively large compared to the UAV size. A larger voxel size causes a higher percentage of voxels to be passed by the sparse rays towards measured points, reducing the gaps.

In this paper, we present an approach to monocular exploration that works by explicitly taking into account the properties of sparse monocular SLAM in both mapping and planning. Our system takes advantage of the potential of mapping-planning coupling, while at the same time producing a semi-dense 3D map of free and occupied space. Thanks to the coupling, the approach is able to explore large-scale unstructured environments, including outdoor areas with large open spaces, which has not yet been demonstrated in literature.

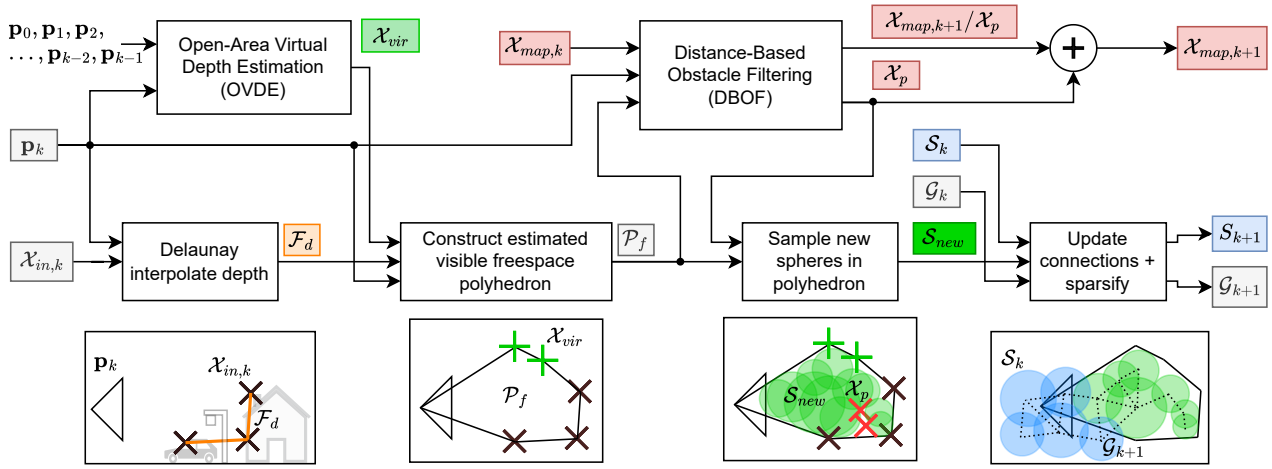

Fig. 2: Mapping pipeline overview. The OVDE and DBOF modules are illustrated in Figure 3 and Figure 4 respectively.

III. Sphere-Based Mapping Using Sparse Visual SLAM Points

Our proposed mapping approach represents free space by a graph of intersecting spheres. Each sphere at the k-th update iteration has a static center $\mathbf{c}$ and changing radius r_k. The radius r_k represents the distance from $\mathbf{c}$ to the nearest unknown space or obstacle at time step k. To allow rapid path planning, we maintain a graph $\mathbf{G}$, which contains an edge for each pair of intersecting spheres. To represent obstacles, we accumulate the stable 3D points (i.e. with low position covariance) estimated by the monocular SLAM into a set of points $\mathcal{X}$ with user-defined minimal point-to-point distance. The obstacle points are updated, merged and removed based on criteria designed specifically for monocular sensing, described in subsection III-C.

The mapping runs onboard the UAV in real-time, using only the estimated pose and 3D points from monocular-inertial SLAM as its inputs. In this paper, we use the sparse point-based OpenVINS [22] method as the SLAM frontend, but our method could work with other monocular-inertial SLAMs. A single map update iteration consists of the following steps, visualized also in Figure 2.

A. Depth Interpolation and Polyhedron Construction

The first step is to construct a polyhedron of space that is likely to contain free space based on the information in the current timestep. We interpolate depth between the sparse SLAM points using a simplified version of the approach described in FLAME [18], where a precise mesh is constructed from visual keypoint measurements. In this paper, we care primarily about mapping the free space for navigation purposes, and thus we do not perform the mesh optimization or split the 2D interpolations as in [18].

We project the currently tracked visual SLAM points into the image plane and compute their Delaunay triangulation. By connecting the points in 3D according to their Delaunay triangulation in 2D, we obtain a mesh $\mathcal{F}_d$ that we call a *depth mesh*, which is used only for the current frame. We estimate the currently visible free space as the volume between the camera's position $\mathbf{p}$ and all points on $\mathcal{F}_d$. This space is enclosed by connecting all points that lie on the edge of $\mathcal{F}_d$ to the camera's focal point, which forms an *estimated visible free space polyhedron* $\mathcal{P}_f$. It is only used for the update step at the current time and discarded afterwards.

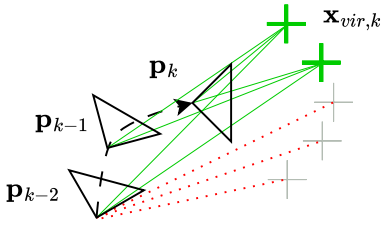

Fig. 3: Open-area virtual-depth estimation diagram. The thick green points $\mathbf{x}_{vir,k}$ satisfy the condition defined in subsection III-B and are added to the construction of $\mathcal{F}_d$

B. Open-Area Virtual-Depth Estimation (OVDE)

When moving in open areas, such as a grassy field without vertical obstacles in Figure 2, point-based visual SLAM can reliably localize only nearby points corresponding to the ground. Using the ray-tracing grid-based mapping method [4], [5], this would cause free space to be estimated only in a downward direction and not allow flight across the field. To allow flight in open areas, our method estimates additional free space based on the following assumption: Consider a virtual point in space $\mathbf{x}_{vir}$ that falls into the camera's FoV at the current pose $\mathbf{p}_k$ and also at previous poses $\mathbf{p}_{k-n}, ..., \mathbf{p}_{k-2}, \mathbf{p}_{k-1}$. If there is sufficient parallax between any two of these poses for estimating the distance of $\mathbf{x}_{vir}$ (given by the max covariance of the measured SLAM points), and if the visual SLAM frontend has not localized any stable 3D point approximately in the direction of $\mathbf{x}_{vir}$, then $\mathbf{x}_{vir}$ must lie in free space (see Figure 3).

The mapping method periodically checks the described condition for a set of virtual points $\mathbf{x}_{vir}$ at fixed positions

relative to the UAV (see Figure 2) at each mapping iteration and for a fixed number of previous poses. The points that fulfill this condition are used to recompute the *depth mesh* $\mathcal{F}_d$ as if they were measured by the SLAM frontend. However, we discard points that would fall into the Delaunay triangulation of the visible obstacle points, since there is a more reliable depth estimate from interpolating the SLAM points. Free space estimated in this manner also cannot erase existing obstacle points in the map.

This approach allows flexibly adding more free space in open areas while using the SLAM points for estimating free space near obstacles. Of course, the assumption does not hold close to obstacles that are not detectable by the sparse visual SLAM frontend (e.g. textureless objects or thin wires). However, this could be resolved by using a semi-dense or dense SLAM frontend (at a higher computational cost), or by adding low-cost ultrasonic sensors to detect close obstacles.

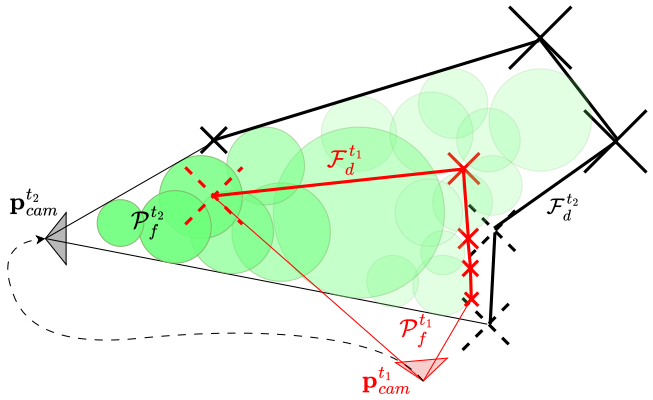

Fig. 4: Top-down illustration of sphere sampling and obstacle point management: The visible free-space polyhedron $\mathcal{P}_f^{t_2}$ is created using the currently tracked SLAM points (black crosses) and the camera's position $\mathbf{p}_{cam}^{t_2}$ at time t_2. The size of a point $\mathbf{x}$ corresponds to the distance $d_\mathbf{x}$ that it was measured from. The (dashed red) point seen far away from the camera at a previous time t_1 now lies in $\mathcal{P}_f^{t_2}$ close to the camera and will thus be deleted as noise. The (dashed black) two points observed far from the camera at t_2 are not added to the map, as there are more precisely localized (red) points close to them. New spheres are inscribed inside $\mathcal{P}_f^{t_2}$, but their radii are also bound by the protected points (non-dashed red) $\mathcal{X}_p$, which were added to the map at t_1.

C. Distance-Based Obstacle Filtering (DBOF)

As the next step of the update, we decide which tracked SLAM points to add into the map points $\mathcal{X}$, and which points already in the map to delete. An important piece of information is the position covariance of each SLAM point. Since in inverse depth parametrization, the position covariance of a point grows with the distance from which it was measured [23], we approximate the covariance simply as the lowest distance $d_{m,x}$ that a point has been observed from. This information is crucial for solving problems that can arise when viewing some surface from a much higher distance than before (e.g. observing a wall from 5 m, and then later from 20m).

As illustrated in Figure 4, points corresonding to a small obstacle might momentarily not be detected by the SLAM frotend, while a larger obstacle behind the small one is detected. Thus, we cannot simply delete every map point that falls into $\mathcal{P}_f$, because then the original points would be wrongly deleted and assumed as free space. We solve this problem by deleting any map point $\mathbf{x}$ that falls into $\mathcal{P}_f$ only if

$$\mathbf{x} \in \mathcal{P}_f \quad \text{and} \quad |\mathbf{x} - \mathbf{p}_{cam}| < \alpha \cdot d_{m,x}, \tag{1}$$

where $\mathbf{p}_{cam}$ is the position of the camera and α is a paremeter, which we empirically set to 1.1 to account for distance measurement noise. The points that lie in the free-space polyhedron $\mathcal{P}_f$ and are not deleted are called *protected points* $\mathcal{X}_p$. These points are used to constrain sphere radii in the following update step.

Additionally, to prioritize closer measurements, points seen from a closer distance replace points seen from larger distances, if they are measured close enough to them. In the same way, new points seen at a large distance are not added to the map, if there are more accurately localized points near them, which is also visualized in Figure 4. We enforce this rule by deleting any newly added or previously mapped point $\mathbf{x}$ if

$$\exists \mathbf{x}_{prev} : |\mathbf{x} - \mathbf{x}_{prev}| < a \quad \text{and} \quad d_{m,x_{prev}} < d_{m,x} \tag{2}$$

where $\mathbf{x}_{prev}$ is another mapped or newly added point with a higher position accuracy and a is the minimal point-to-point distance set by the user.

D. Updating Existing Spheres

Next, we recompute the radii of spheres that could be updated by $\mathcal{P}_f$ or the input points $\mathcal{X}_{in}$. To bound the update time of this step, we check the largest sphere radius r_{max} in the map, which allows us to quickly filter out all spheres whose centers fall outside a bounding box around $\mathcal{P}_f$, inflated by r_{max}. Then, for any remaining sphere with a center $\mathbf{c}$ and radius r_k, the updated radius is computed as

$$r_{k+1} = \min\left(\max\left(r_k, d(\mathbf{c}, \mathcal{P}_f)\right), d(\mathbf{c}, \mathcal{X}_{in} \cup \mathcal{X}_p)\right). \tag{3}$$

In this equation, $d(\mathbf{c}, \mathcal{P}_f)$ is the signed distance to $\mathcal{P}_f$ (positive if the point is inside the polyhedron, negative if outside). The value of $d(\mathbf{c}, \mathcal{X}_{in} \cup \mathcal{X}_p)$ is the minimum distance to all input obstacle points $\mathcal{X}_{in}$, and to the protected map points $\mathcal{X}_p$ described in subsection III-C. Essentially, this equation means that the polyhedron $\mathcal{P}_f$ is used to increase the radius of a sphere, whereas the protected $\mathcal{X}_p$ and input points $\mathcal{X}_{in}$ are used to constrain it. As the last part of this step, we delete all spheres with $r_{k+1} < r_{min}$, where r_{min} is the smallest allowed sphere radius specified by the user.

E. Sampling New Spheres

To introduce new spheres into the map, we sample a fixed number of points inside $\mathcal{P}_f$ at random distances between the camera and the depth mesh F_o. A potential new sphere's

radius is determined in the same way as for the old spheres in the previous step in Eq. 3 with $r_k = 0$. If the potential radius is larger than r_{min}, the sphere is added to the map.

F. Recomputing and Sparsifying Sphere Graph

After all the sphere radii updates have been made, we update the graph of spheres used for path planning, so that all intersecting spheres are connected in the graph. Furthermore, to constrain map update time and path planning time, we perform a redundancy check on the updated and added spheres in the same way as in [10]. If any sphere is found to be redundant, it is deleted from the map. This step is important to cover large open areas by only a few spheres to allow rapid planning, and tight corridors by a higher density of spheres, capturing the information about potential paths and distances in more detail.

IV. Perception-Aware Exploration with a Sparse Sphere-Based Map

In this section, we describe a monocular exploration approach enabled by the proposed mapping method. In principle, we employ the commonly used next-best-view (NBV) [24] strategy, which has so far been deemed unsuitable for monocular exploration in previous works [4], [5]. In the following sections, we introduce several major differences in methodology that are critical for allowing the NBV strategy to be used for robots equipped with only a monocular camera for depth sensing.

A. Frontier Sampling on Free-Space Polyhedron

Firstly, since the proposed map representation is fundamentally different from an occupancy grid, we need to detect frontiers (i.e. the boundary of free and unknown space) in a different way than with an occupancy grid. We propose to sample points along the visible free-space polyhedron described in section III at each map update, and add the points as frontiers, if they do not lie inside any free-space sphere and if they are at a user-defined distance from all map obstacle points and other frontiers. If they do not meet these criteria in any following update, they are deleted.

In enclosed spaces, which are usually considered in autonomous exploration literature, all frontiers are usually worth exploring. However, in outdoor exploration with areas of open space, it is also important to assign different value to different frontiers, especially for UAVs. We propose to constrain the exploration to only consider frontiers that lie near some obstacle points. This way, the UAV does not explore up into the sky. Additionally, this forces the UAV to only explore near texture-rich areas, thus avoiding losing visual SLAM tracking. These frontier points are used to generate exploration viewpoints such that at least some frontier points are visible form each viewpoint.

B. Forced Translation when reaching Viewpoints

In traditional exploration approaches with dense distance sensors, it is sufficient to move the robot to a frontier, aim the sensors towards the unknown space, and assume that the distance sensors will uncover some additional space behind the frontier and thus expand the map. This approach will not, in principle, work reliably with a robot using a monocular camera for estimating depth from motion. Such a robot requires *translational* motion to obtain correct distance estimates of visual keypoints. With rotation only, or motions that have too much rotation compared to translation, reliable motion-based depth estimates cannot be obtained.

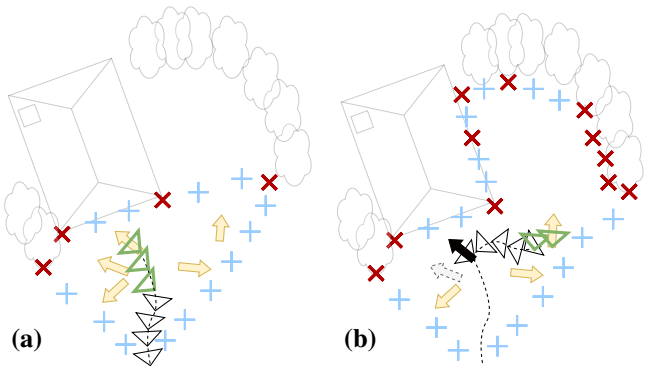

Fig. 5: Top-down illustration of the exploration approach. In (a), the UAV navigates to one of the sampled viewpoints (yellow) aimed at a feature-sparse wall. This does not uncover any frontiers (magenta), so to stop returning to this or nearby viewpoints, the viewpoint is blocked (black). In both (a) and (b), the UAV flies facing forward and aligns itself with the viewpoint's direction when nearing it (green poses).

To solve this, we propose a simple perception-aware planning approach: When finding a path to an exploration viewpoint, we force the trajectory to aim the UAV in the viewpoint's direction after getting to a pre-defined distance d_c from the goal, as visualized in Figure 5. This way, the UAV reaches the viewpoint with at least d_c meters of translational motion. If textured surfaces are visible from that viewpoint, the UAV will most likely observe sufficient parallax to estimate their distances.

C. Explored Viewpoint Blocking

Another necessary distinction we propose for NBV exploration with a monocular camera is to block sampling of new exploration viewpoints near visited viewpoints. This is due to the fact that real-world environments often contain textureless areas (mainly in buildings or other man-made structures). Since obstacle points correspond to textured points localized by the visual SLAM, no points will ever be added on textureless surfaces. Frontier points will be generated there instead, since the surfaces lie on the edge of the visible free space polyhedron and can be far enough from any obstacle points. However, such frontiers cannot uncover any new space, since there is nothing behind them. For this reason, we block the sampling of new viewpoints near visited viewpoints, visualized as large black arrows in Figure 6 This stops the UAV from getting stuck repeatedly trying to uncover such surfaces.

D. Safety-Aware Planning

Lastly, the UAV is controlled to always fly in the direction of its camera when not aligning itself with a viewpoint as described in subsection IV-B. This approach is crucial for ensuring collision-free flight in the case of incorrectly initialized free space. As discussed in subsection V-C, some obstacles (e.g. thin wires, twigs) might not be visible in the cameras from a large distance to be detected by the visual SLAM (especially when using fisheye cameras), and can be wrongly estimated as free space. Using the forward-facing flight, the UAV has a higher chance to see these obstacles that were previously not detected, and quickly replan. To further increase safety and allow agile maneuvers, the paths are planned with a strong preference for high distance from obstacles, for which the graph-of-spheres representation is highly suitable, as described in [9]–[11].

V. Experiments

In this section, we analyze the performance of the proposed method in large-scale real-world (Sec. V-A) and simulated (Sec. V-B) environments. The presented implementation of the proposed mapping and exploration methods is currently single-threaded and written in python. With this implementation, the mapping runs on average at $4.5\,\mathrm{Hz}$ on average on a standard CPU. This is sufficient for exploration planning, but can be slow to detect obstacles at high speeds. For this reason, the UAV also checks the predicted trajectory for potential collisions with the input pointclouds at a higher rate.

The UAV used in the experiments was equipped with a monocular global-shutter grayscale fisheye Bluefox camera, and ICM-42688 IMU. The MRS UAV system [25] was used for low-level motion planning, control, and the grid-based path planning for the simulation comparisons. For state estimation and as the source of sparse visual SLAM points, we used OpenVINS [22] in the inverse-depth measurement mode. (redundant? -¿ needto specify assumption at start) However, any other monocular-inertial SLAM could be used as the fronted of our method, as long as it tracks points approximately uniformly in the camera's FOV. This property is necessary to assure the assumption that small objects will be detected at a close range, as discussed in subsection III-C.

A. Real-World UAV Monocular-Inertial Exploration

In Figure 6, we present an $8\,\mathrm{min}$ experiment where our approach is validated in real-world conditions. We bounded the area for generating exploration goals to a 70x10x8 m box around the side of an abandoned farmhouse, and the UAV fully autonomously explored nearly all the available space in this region. The UAV successfully avoided and mapped all the debris, bushes and the house walls in the area, and when battery levels were getting low, it autonomously returned to the starting position. Also note that the UAV mapped a considerable amount of space in the open field to the left of the house. This was made possible by our approach to safe estimation of free space in open-area areas, described in subsection III-B.

We have also tested the proposed approach in a variety of other, smaller real-world environments, including an orchard, which our method fully autonomously explored. Videos from these real-world experiments are available in the multimedia materials.

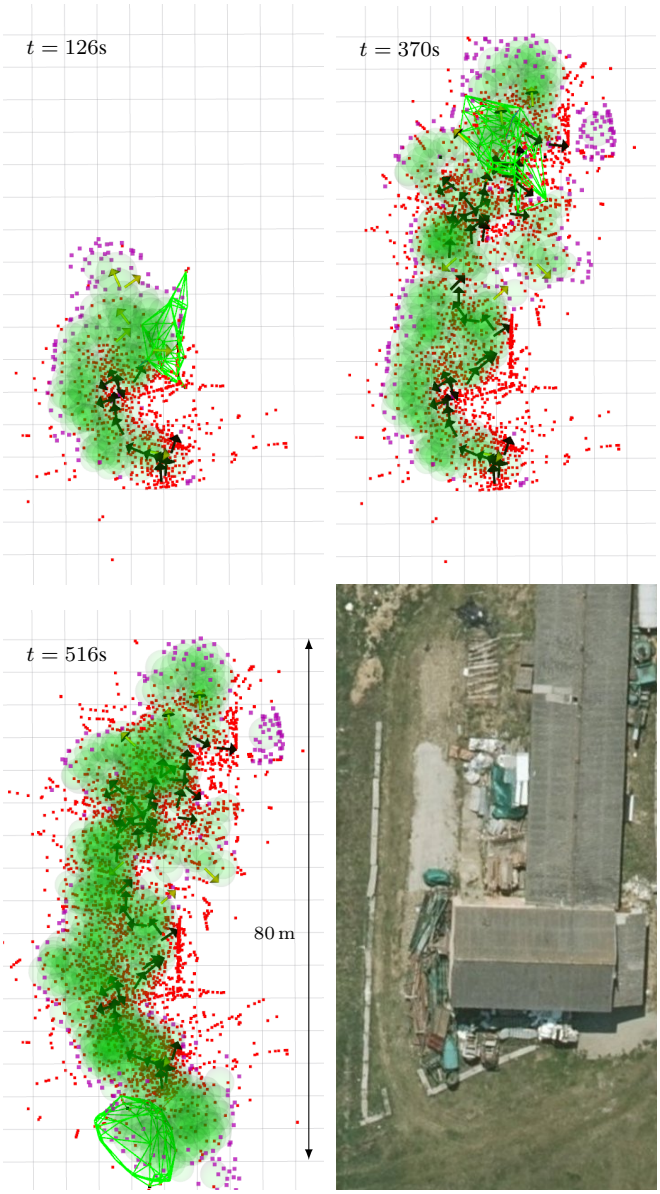

Fig. 6: Visualization of the real-world autonomous exploration experiment described in subsection V-A, along with a satellite image of the explored abandoned farm area. Free space (green) is shown alongside mapped obstacle points (red), frontiers (purple) and explored viewpoints (black).

B. Simulation Experiments

Q: Does the mapping and planning enable exploration at larger scales than conventional approaches? As the previously published monocular exploration methods do not have

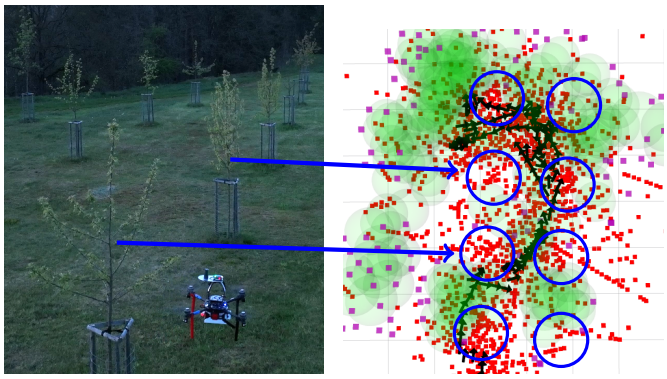

Fig. 7: The 4min orchard exploration experiment. The map is noisy due to low-light conditions, trees are highlighted in blue for clarity.

Method	World	Mean A	Max A	Runs
MonoSpheres	Urban	4919 m^2	6081 m^2	6
Grid-Based	Urban	1911 m^2	2306 m^2	6
No FFF	Urban	2470 m^2	3181 m^2	3
No DBOF	Urban	2308 m^2	3081 m^2	3
No OVDE	Urban	747 m^2	862 m^2	3
No VPB	Urban	3697 m^2	4175 m^2	3
MonoSpheres	Cave	7627 m^2	8006 m^2	3
Grid-Based	Cave	3171 m^2	3375 m^2	3
MonoSpheres	Rooftops	3760 m^2	4606 m^2	3
Grid-Based	Rooftops	1360 m^2	1556 m^2	3
MonoSpheres	Sparse	5127 m^2	5150 m^2	3
Grid-Based	Sparse	2108 m^2	2168 m^2	3

TABLE I: Quantitative results showing the explored area in the simulation experiments described in subsection V-B.

code available (with the exception of [6], which, however, is not ROS-compatible as of writing this paper), we prepared a simple grid-based mapping and exploration pipeline for comparison with MonoSpheres. This reference method uses the same mapping approach as [4], [5] (i.e. initializing free space between the camera and the visual SLAM points that have a low position covariance). To explore, the method periodically samples free-space points uniformly in a 40x40x10 m area around the robot and sends these as goals to a grid-based A* path planning module, which finds path through the occupancy grid that would move the robot as close as possible to the goal while respecting the same distance from obstacles and uknown space as the MonoSpheres method ($1.5\,\mathrm{m}$ in these experiments). The robot then follows these paths with the camera pointing forward and replans at $5\,\mathrm{Hz}$ to handle measurement inacurracies. We use OctoMap [13] for the grid-based mapping with a $0.5\,\mathrm{m}$ cell size. This value was chosen because smaller cell sizes caused the robot to not find any safe paths due to the free-space gaps documented in [4] and larger cell sizes blocked exploration of narrow areas.

As the evaluation metric, the volume of the constructed map is commonly used in exploration literature. However, because of the open-area free-space sampling scheme and the depth interpolation, MonoSpheres constructs considerably more free space than the raytracing mapping approach over the same trajectories. For this reason, the volume metric would not be fair towards methods using the grid-based raytracing mapping. We instead divide the target exploration area into 2.5x2.5m columns that are marked as explored if the constructed map (sphere-based or grid-based) contains any element in a given column. The metric, A, is then the top-down 2D area of all the explored columns.

We compare the grid-based approach with MonoSpheres in multiple experiments on 2 different worlds, and terminate each run after $20\,\mathrm{min}$ or sooner if a collision occurs. The resulting maps and exploration progress are shown in Figure 9, **??** and the mean and max explored area in Table I. Videos from these experiments are available in the multimedia materials. The experiments were conducted in the Gazebo simulator [26] with the same UAV platform and sensory setup as in the real-world experiments. Exploration goal generation was constrained to a 60x50x7 m area of interest in the urban world for both methods, and to 170x110x20 m in the cave world.

C. Discussion

In the "cave" world, which is a texture-rich indoor environment, the grid-based approach performed similarly to MonoSpheres in terms of initial exploration speed. With a more sophisticated viewpoint selection approach, the grid-based approach could explore as much area in the end as MonoSpheres, supporting the viability of this mapping approach in indoor, textured areas claimed in [4], [5].

However, in the "urban" world – an open-space, texture-sparse environment – the grid-based approach was never able to explore a large portions of the area that require the UAV to fly over a building or move across an open field. In comparison, MonoSpheres (with the same parameters as in the cave world) was able to consistently cover the outdoor environment. These results demonstrate that our proposed open-space free-space sampling approach described in subsection III-B enables exploration of outdoor areas, where the ray-tracing grid-based mapping is not able to create sufficient free space, while also safely mapping indoor areas.

In all of the experiments, our approach achieved long periods of collision-free exploration even near complex obstacles such as trees, collapsed houses, and cars in uncommon configurations. In the cave world with objects and walls rich in texture, neither of the compared methods collided with the environment. Thus, we designed the urban environment to test the limits of our method, making it contain textureless areas (e.g. cars and burned parts of buildings) and thin obstacles such as exposed metal rods. As expected, this led to collisions in both methods, as these objects cannot be detected by the used OpenVINS frontend. In particular, the grid-based method collided once with a smooth metal rod, and the MonoSpheres method collided in two runs with a car and with the burned textureless part of a building, albeit after exploring the majority of the environment. This is expected, as these objects cannot be detected by the used OpenVINS frontend, and could be improved by using a visual front-end that uses line-based or dense features (at the

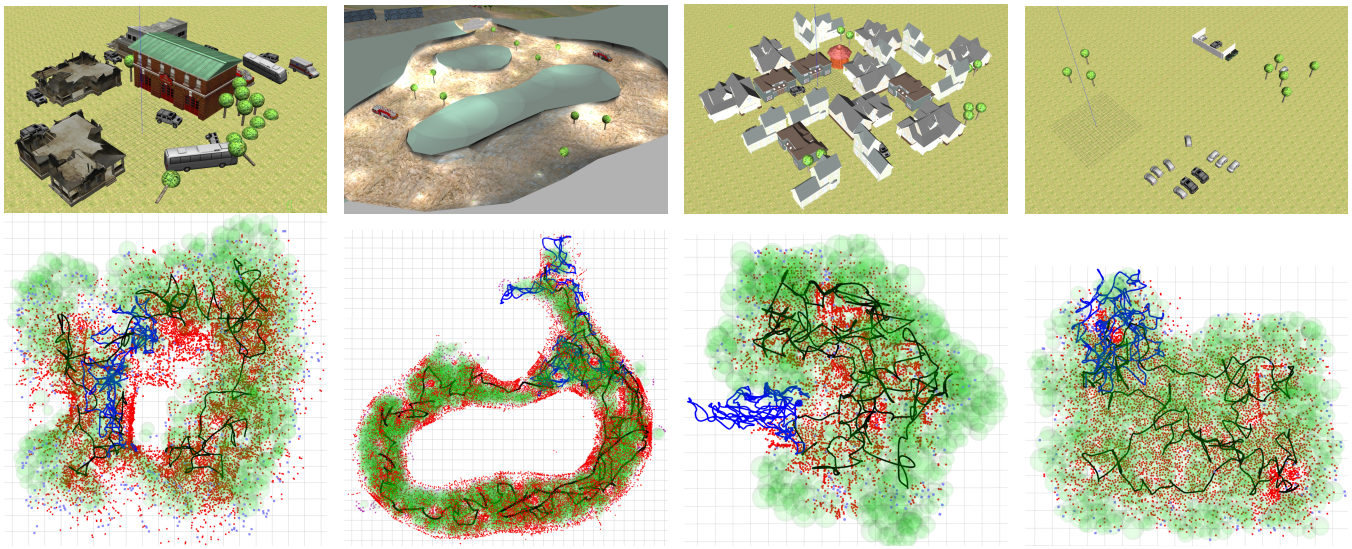

Fig. 8: Top simulated runs of MonoSpheres with MonoSpheres (black) and Grid-Based explorer (blue) trajectories

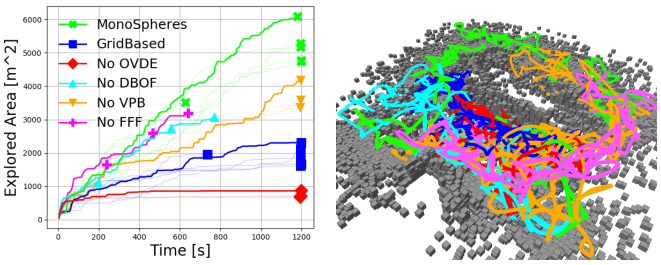

Fig. 9: Ablation test results on the Urban world

cost of computational complexity), or by leveraging low-cost ultrasound sensors as a bumper for the UAV.

VI. CONCLUSION

In this paper, we presented a novel, perception-coupled approach to mapping and planning for UAV exploration of 3D space with a monocular camera. Ablation experiments confirm the utility of the individual modules of our proposed method. Compared to the existing approaches to 3D monocular exploration, which have so far presented autonomy at the scale of a few indoor rooms [4]–[6], our proposed approach achieves autonomous exploration at the scale of an urban area with multiple buildings, trees and wide open spaces. With the same configuration, our method allows exploration of various environment topologies – sparse, dense, indoor, outdoor and environments requiring flight over obstacles in wide open spaces. In future work, our method could be improved by adopting a submap-based approach or by incorporating inexpensive ultrasonic sensors to act as "bumpers" against thin objects that are hard to detect by cameras.

REFERENCES

[1] M. Tranzatto, T. Miki, M. Dharmadhikari *et al.*, "CERBERUS in the DARPA Subterranean Challenge," *Science Robotics*, vol. 7, 2022.

[2] N. Hudson, F. Talbot, M. Cox *et al.*, "Heterogeneous Ground and Air Platforms, Homogeneous Sensing: Team CSIRO Data61's Approach to the DARPA Subterranean Challenge," *Field Robotics*, vol. 2, pp. 595–636, 2021.

[3] M. Petrlík, P. Petráček, V. Krátký *et al.*, "UAVs Beneath the Surface: Cooperative Autonomy for Subterranean Search and Rescue in DARPA SubT," *Field Robotics*, vol. 3, pp. 1–68, 2022.

[4] L. von Stumberg, V. C. Usenko, J. J. Engel *et al.*, "From monocular SLAM to autonomous drone exploration," *2017 European Conference on Mobile Robots (ECMR)*, pp. 1–8, 2016.

[5] D. Pittol, M. Mantelli, R. Maffei *et al.*, "Monocular 3D Exploration using Lines-of-Sight and Local Maps," *Journal of Intelligent & Robotic Systems*, vol. 100, pp. 465 – 481, 2020.

[6] N. Simon and A. Majumdar, "MonoNav: MAV Navigation via Monocular Depth Estimation and Reconstruction," in *Symposium on Experimental Robotics (ISER)*, 2023.

[7] A. Steenbeek and F. Nex, "CNN-Based Dense Monocular Visual SLAM for Real-Time UAV Exploration in Emergency Conditions," *Drones*, 2022.

[8] J. Gaigalas, L. Perkauskas, H. Gricius *et al.*, "A Framework for Autonomous UAV Navigation Based on Monocular Depth Estimation," *Drones*, 2025.

[9] M. Spencer, R. Sawtell, and S. Kitchen, "Sphere-Graph: A Compact 3D Topological Map for Robotic Navigation and Segmentation of Complex Environments," *IEEE Robotics and Automation Letters*, vol. 9, pp. 2567–2574, 2024.

[10] T. Musil, M. Petrlík, and M. Saska, "SphereMap: Dynamic Multi-Layer Graph Structure for Rapid Safety-Aware UAV Planning," *IEEE Robotics and Automation Letters*, vol. 7, pp. 11 007–11 014, 2022.

[11] Y. Ren, F. Zhu, W. Liu *et al.*, "Bubble Planner: Planning High-speed Smooth Quadrotor Trajectories using Receding Corridors," *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6332–6339, 2022.

[12] H. P. Moravec and A. Elfes, "High resolution maps from wide angle sonar," *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, vol. 2, pp. 116–121, 1985.

[13] A. Hornung, K. M. Wurm, M. Bennewitz *et al.*, "OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees," *Autonomous Robots*, 2013.

[14] H. Oleynikova, Z. Taylor, M. Fehr *et al.*, "Voxblox: Incremental 3D Euclidean Signed Distance Fields for on-board MAV planning," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1366–1373, 2017.

[15] D. Duberg and P. Jensfelt, "UFOMap: An Efficient Probabilistic 3D Mapping Framework That Embraces the Unknown," *IEEE Robotics and Automation Letters*, vol. 5, pp. 6411–6418, 2020.

[16] Y. Ming, X. Meng, C. Fan *et al.*, "Deep learning for monocular depth estimation: A review," *Neurocomputing*, vol. 438, pp. 14–33, 2021.

[17] U. Rajapaksha, F. Sohel, H. Laga *et al.*, "Deep Learning-based Depth Estimation Methods from Monocular Image and Videos: A Comprehensive Survey," *ACM Comput. Surv.*, vol. 56, no. 12, Oct. 2024.

[18] W. N. Greene and N. Roy, "FLaME: Fast Lightweight Mesh Estimation Using Variational Smoothing on Delaunay Graphs," *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 4696–4704, 2017.

[19] W. G. A. Castillo, V. P. Casaliglla, and J. L. Pólit, "Obstacle Avoidance Based-Visual Navigation for Micro Aerial Vehicles," *Electronics*, vol. 6, p. 10, 2017.

[20] K. Celik, S.-J. Chung, M. Clausman *et al.*, "Monocular vision SLAM for indoor aerial vehicles," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009, pp. 1566–1573.

[21] 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*, pp. 59–66, 2017.

[22] P. Geneva, K. Eckenhoff, W. Lee *et al.*, "OpenVINS: A Research Platform for Visual-Inertial Estimation," *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4666–4672, 2020.

[23] J. Civera, A. J. Davison, and J. M. M. Montiel, "Inverse Depth Parametrization for Monocular SLAM," *IEEE Transactions on Robotics*, vol. 24, pp. 932–945, 2008.

[24] 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'*, 1997, pp. 146–151.

[25] T. Baca, M. Petrlik, M. Vrba *et al.*, "The MRS UAV System: Pushing the Frontiers of Reproducible Research, Real-world Deployment, and Education with Autonomous Unmanned Aerial Vehicles," *Journal of Intelligent & Robotic Systems*, vol. 102, no. 26, pp. 1–28, May 2021.

[26] N. P. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, pp. 2149–2154 vol.3, 2004.