# Monocular-Inertial UAV Exploration and Mapping of Large-Scale Outdoor Environments using Sparse Visual SLAM

Tomáš Musil, Matěj Petrlík, Martin Saska

***Abstract*— We present a novel approach to mapping 3D space for navigation purposes, using only pose estimates and keypoints from sparse monocular-inertial SLAM as input. We propose to represent an environment by a graph of intersecting free-space spheres and a set of obstacle points with remembered measurement distance, which is fundamentally different from traditionally used grid-based maps. A polyhedron of approximated visible free-space is formed at each map iteration from tracked SLAM points and the camera's position. This polyhedron is used for updating sphere radii and deleting obstacle points. Furthermore, we describe an exploration approach that uses ~~the advantages of~~ the novel representation to overcome the challenges of monocular vision-based UAV autonomy. The presented method allows large-scale autonomous exploration in unstructured indoor/outdoor 3D environments, using only a single monocular camera and an IMU onboard a possibly micro-scale UAV. We open-source the code for our methods to provide 3D mapping and exploration capabilities to any UAV with visual-inertial state estimation.**

**Code— TODO**

## I. INTRODUCTION

The autonomous UAV systems operating in unknown environments documented in recent literature [1], [2] usually construct an occupancy grid [3], such as the efficient octree-based implementation of OctoMap [4], and use it as ~~the base~~ representation for planning, exploration and constructing higher-level spatial abstractions [5], [6]. However, building a detailed occupancy grid requires dense pointclouds of range measurements due to the raycasting nature of occupancy grid updates. Thus, state-of-the-art exploration and navigation UAV systems have been constrained to expensive, heavy or limited-range sensors such as LiDARs, depth cameras or stereo-camera pairs, which can provide these dense point clouds.

Building an occupancy representation for exploration using only a monocular camera for depth sensing is challenging for many reasons, and has so far been demonstrated only in small-scale indoor environments. However, being able to build a representation that would allow safe and rapid navigation in unknown environments on-board UAVs with such small, light-weight and inexpensive sensors, would unlock many potential applications, such as swarms of disposable UAVs for search and rescue missions. In addition, cameras are ~~useful~~ for object recognition and other types of vision-based perception beneficial for autonomous UAVs, and thus they are already equipped on UAVs in most real-world applications. Furthermore, a single camera onboard a moving UAV can ~~theoretically~~ provide rough distance estimates on distant objects when using the inverse depth parametrization [7], ~~far beyond the approx. 50m range of~~ LiDARs.

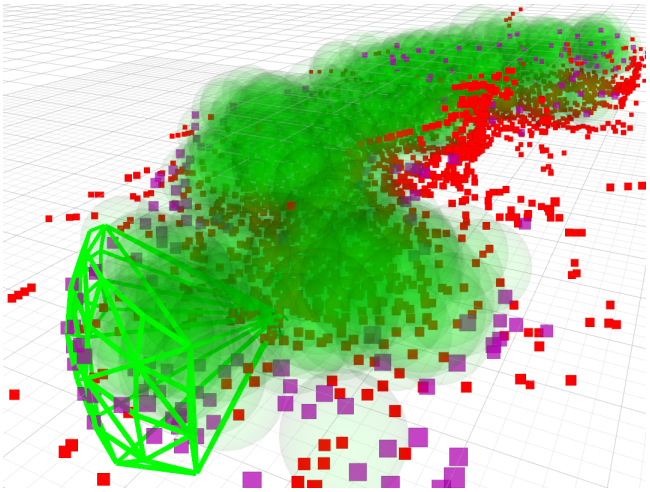

Fig. 1: Visualization of the proposed map representation and how it is updated. The green polyhedron is an estimate of currently visible free space and is formed by the visible triangulated keypoints and the camera's position (axes). The mapped spheres (green), obstacle points (red) and frontier points (purple) are updated by the polyhedron and visible keypoints, as described in section II

In this paper, we present a novel approach to 3D mapping that takes in a sparse pointcloud of visual keypoints tracked by monocular SLAM (already a necessary part of many vision-based UAV systems). Instead of traditionally casting rays through a grid and updating its cells, we construct a polyhedron of approximate visible free space at each frame, inscribe spheres of free space inside that polyhedron, and connect them together to form a graph of intersecting spheres. ~~By storing obstacles~~ as a set of points, we can assign additional metrics to each point, such as measurement distance, ~~or, in future work, the modality of its origin. This~~ allows for rule-based measurement fusion, which is critical for dealing with monocular cameras, as we explain in subsubsection II-B.2. We further show how the proposed mapping method enables autonomous exploration in real-world ~~and simulated~~ large-scale outdoor environments on a UAV using only a single camera and IMU as its sensors, which, to the authors' best knowledge, has not yet been ~~achieved~~ in literature.

Authors are with the Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague, 166 36 Prague 6, {musilto8|matej.petrlik|martin.saska}@fel.cvut.cz

Digital Object Identifier (DOI): see top of this page.

### A. Related Works – Spatial Representations

~~In robotic~~ exploration, the majority of approaches use a grid-based representation of occupied, unknown, and free space, either in the form of an euclidean signed distance field (ESDF) [8] or an occupancy grid [3], [4]. A grid-based representation, while being the basis for a substantial amount of robotic path planning and higher-level spatial abstraction methods [5], [6], [9], has several drawbacks for usage on UAVs without dense range sensors in varying-scale environments.

Firstly, both ESDFs and occupancy grids are most commonly updated by casting rays towards measured 3D points and updating the values of traversed cells. Using this technique with sparse points from monocular SLAM, e.g. in texture-poor environments, can lead to gaps in the map. Existing works address this issue in multiple ways: In [10], the authors take points from semi-dense monocular SLAM and raycast towards them to build an OctoMap [4] occupancy grid, but only demonstrate their local exploration approach built on top of this mapping pipeline to work inside a single room with textured walls. The authors of [11] use sparse monocular SLAM, they also cast rays towards the measured points to build an occupancy grid, and show exploration on a UAV across multiple rooms. They set the occupancy grid cell size to be 0.5m, which is relatively large compared to the UAV size. Such rough voxel size cannot represent the distance to obstacles in high resolution, and if the grid is misaligned with the building, narrow passages can be represented as non-traversible.

This is the second fundamental problem of grid-based representations – the smallest-voxel size must be set by the user. If this is set as too large, narrow passages will not be captured in the map, and if it is too small, the map will need large amounts of raycasts to set all the cells' occupancy values, which ~~is computationally expensive~~. In our previous work [6], we ~~have shown~~ that representing free space by a graph of intersecting spheres can efficiently describe large-space while keeping detailed information in narrow spaces. In additon, it allows orders of magnitude faster path planning than grid-based representations. In ~~this paper~~, we represent free space using spheres as well, but with two major differences:

- In [6], the method required building a local occupancy grid with a LiDAR for updating the spheres. In ~~this paper,~~ we build the graph of spheres *directly* out of 3D measurement points without any intermediate occupancy grid, and only require a monocular camera running visual SLAM instead of a LiDAR.
- Additionally, ~~the~~ representation ~~proposed in this paper stores~~ visual keypoints as ~~obstacle points and remembers their measurement distance~~. These points are important for correctly ~~estimating free space~~ when previously seen obstacles are momentarily not detected by the visual SLAM (see II-B.2).

### B. Related Works – Vision-Based Exploration

Vision-based UAV autonomous exploration has not yet reached the levels of autonomy as when using dense depth sensors, and remains a challenge, as documented in recent surveys [12], but some progress has been ~~made.~~ In [13], the authors presented the first ever vision-based UAV system that can explore and build a map (a 3D occupancy grid) on a UAV equipped with stereo cameras and also a supporting downward-facing optical flow sensor for better motion estimation. The authors used dense pointclouds coming from the stereo cameras for building an OctoMap [4] occupancy representation in the same way as with depth data from an RGB-D camera or LiDAR. This allowed them to use similar exploration techniques as on systems with such sensors.

For UAVs equipped with only a monocular camera, to the authors' best knowledge, all the existing works have demonstrated autonomous 3D mapping and exploration on a UAV only indoor and at the small scale of a single room [10], [14] or a few simple rooms [11], [15]. Monocular SLAM, both sparse and dense, cannot produce measurement points on textureless areas, and the existing approaches either accept that the data from visual SLAM can be very sparse, or they use deep-learning-based depth estimation to obtain dense data.

The authors of [10], [11] argue against using frontier-based methods [16], due to the facts that no points will ever be generated on textureless areas, and thus those frontiers would not be uncovered by a monocular camera. The authors have presented alternative exploration approaches, but they have only shown them to work in small-scale environments. In our exploration approach, thanks to not using raycasting in the mapping pipeline and also blocking explored viewpoints, we show that frontier-based exploration is possible and assures global coverage of the explored environments.

In ~~a different approach, the authors of~~ [14], [15] ~~use~~ deep-learning-based single image depth estimation to obtain dense point clouds from a monocular camera, essentially turning it into a depth camera. However, using learning-based monocular depth estimation models brings additional problems. As discussed in recent surveys [17], [18], this approach struggles to provide real-time onboard performance and it is not assured that it will work reliably in domains that the network was not trained on. In addition, a GPU significantly raises the cost and weight of a UAV.

Compared to the above mentioned approaches, we demonstrate that our approach achieves exploration and navigation in 3D using a monocular camera and an IMU in outdoor, large-scale environments, with all of the computations running on-board the UAV, without requiring a GPU.

### C. Contributions

To enable robust vision-based autonomous ~~navigation~~ in unknown, unstructured 3D environments ~~for inexpensive UAVs~~, our paper brings the following contributions:

1) A novel 3D mapping approach~~, which~~ construct~~s~~ a sphere-based map using sparse pointclouds obtained

from monocular-inertial SLAM as the only depth information.
2) ~~A system that demonstrates l~~large-scale 3D outdoor exploration on a UAV equipped with only a monocular camera and IMU ~~as its sensors~~, enabled by the proposed mapping method and validated in real-world conditions.
3) Open-sourced code for the novel mapping and exploration approach, along with example simulation scripts for replicating our results.

## II. Sphere-Based Map Built From Sparse Visual SLAM Keypoints

This section describes the proposed mapping approach. In subsection II-A, we introduce the novel spatial representation and its properties. In subsubsection II-B.1, we construct an estimated visible free-space polyhedron using motion information and SLAM points available at a given time. In subsubsection II-B.2, the free-space polyhedron and keypoint measurement distances are used to determine which points are added or deleted from the map. Next, in subsubsection II-B.3, we describe how the free-space polyhedron is used to update the radii of existing and newly sampled free-space spheres. As the last step, in subsubsection II-B.5, we update the connections of the modified spheres and prune the sphere graph to keep it sparse for rapid path planning.

### A. Map Representation and Notation

We propose to represent 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 to the nearest unknown space and obstacles at $\mathbf{c}$. We maintain a graph $\mathbf{G}$, which contains an edge for each pair of intersecting spheres, and use it for path planning. Furthermore, we maintain a set of obstacle points $\mathbf{X}$, which correspond to the textured surfaces measured by the visual SLAM. For each obstacle point $\mathbf{x} \in \mathbf{X}$, we store the lowest distance that the point was measured from $d_{\mathbf{x},\min}$. A lower minimal measurement distance of a visual keypoint means that the point is less likely to be noise and we make it more difficult to erase it from the map. This is a necessary feature for safe flight when using sparse visual SLAM keypoints as the only depth information, as explained more in subsubsection II-B.2.

### B. Map Update Algorithm

The mapping runs on-board the UAV in real-time, using only the pose and 3D landmark information from visual SLAM as its inputs, approximately at 5Hz. A single map update iteration consists of the following steps in order:

*1) Constructing the Visible Free Space Polyhedron:* The first step is to construct a polygon of space that is estimated to contain free space based on the information in the current timestep. To interpolate depth between the sparse keypoints, we employ a simplified version of the approach described in FLAME [19]. There, the authors focus on constructing a precise mesh from visual keypoint measurements, but here, we care primarily about mapping the free space, for planning purposes, and thus we do not perform the mesh optimization or splitting the 2D interpolations that are done in [19].

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 $\mathbf{F}_d$ 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 focal point and all points on $\mathbf{F}_d$. This space is enclosed by connecting all points that lie on the edge of $\mathbf{F}_d$ to the camera's focal point, which forms an *estimated visible free space polyhedron* $\mathbf{P}_f$, visualized in Figure 1.

When moving in open areas, such as over a field, visual SLAM often tracks only a few keypoints on the ground, which would cause free space to be estimated only below the UAV and not allow flight across the field. To allow safe flight in open areas, we make the following assumption: Consider a virtual keypoint $\mathbf{x}_{vir}$ at time $t_2$ that falls into the UAV's FoV since $t_1$ in the past ~~up to~~ $t_2$ (i.e. it could be tracked by visual SLAM). Then, if there is a point on the UAV's trajectory between $t_1$ and $t_2$ such that there would be enough parallax for estimating the 3D position of $\mathbf{x}_{vir}$ (according to a user-defined threshold), and if the visual SLAM has not triangulated any point at approximately that position, then $\mathbf{x}_{vir}$ must lie in free space. This approximation of course fails ~~in the case of~~ featureless walls, ~~but works quite well in~~ large-scale outdoor environments.

We periodically check the above condition for a set of virtual points $\mathbf{x}_vir$ at a fixed distance in front of the UAV at each iteration. We take the points that fulfill this condition and add them to the creation of the *depth mesh* $\mathbf{F}_d$, as shown in TODO-FIG. However, we discard points that would fall into the Delaunay triangulation of the true obstacle points, since there we have an existing depth estimate. This way, we can safely estimate additional freespace, in the case of sufficiently textured wide outdoor areas.

*2) Updating Obstacle Points:* As the next step of the update, we decide which tracked SLAM points ~~to add~~ into the map points $\mathbf{X}$, and which points in the map ~~to delete~~. An important part of ~~our~~ method is that we store the minimum of the distances that any point $\mathbf{x}$ has been observed from, denoted further as $d_{m,x}$. An apparent problem 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). A situation can occur where the previously mapped points are now triangulated at an incorrect position, or are not triangulated by the visual SLAM at all (e.g. due to insufficient parallax), while some other points that lie beyond the original points are triangulated, as shown in Figure 2. Thus, we cannot simply delete every map point that falls into $\mathbf{P}_f$, because then the original points (red in Figure 2) would be wrongly deleted and assumed as free space.

We solve this problem by deleting any map point $x$ that falls into $\mathbf{P}_f$ only if

$$|\mathbf{x} - \mathbf{p}_{cam}| < 1.1 \cdot d_{m,x} \tag{1}$$

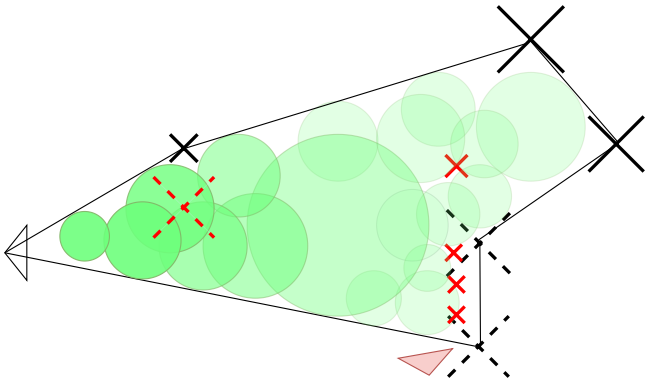

Fig. 2: Diagram of sphere sampling and map point management: Visible free-space polygon (-); Newly sampled spheres (O); Triangulated keypoints visible at current frame (X); Points already in the map (X) added from a previous pose (red). Smaller points and higher-opacity spheres have been observed from lower distances. Dashed crosses correspond to points that will be deleted (red) or not added to the map (black).

where $p_{cam}$ is the position of the camera. The points that lie in $\mathbf{P}_f$ and are not deleted are called *protected points* and are used to constrain sphere radii in the following update step. Additionally, to prioritize closer measurements, points seen from a closer distance can replace points seen at 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 measured points near them, ~~also~~ visualized in Figure 2.

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

$$r_{k+1} = \min\left(\max\left(r_k, d(\mathbf{c}, \mathbf{P}_f)\right), d(\mathbf{c}, \mathbf{X}_{in} \cup \mathbf{X}_p), r_{max}\right), \tag{2}$$

where $d(x, \mathbf{P}_f)$ is the signed distance to $\mathbf{P}_f$ (positive if the point is inside the polyhedron, negative if outside) and $d(\mathbf{x}, \mathbf{X}_{in} \cup \mathbf{X}_p)$ is the minimum distance to all input obstacle points $\mathbf{X}_{in}$, and to the protected map points $\mathbf{X}_p$ described in subsubsection II-B.2. Finally, we delete all spheres with $r_{k+1} < r_{min}$, where $r_{min}$ is the smallest allowed sphere radius specified by the user.

*4) Sampling New Spheres:* To introduce new spheres into the map, we sample a fixed number of points inside $\mathbf{P}_f$ at random distances between the camera and the depth mesh $F_o$. ~~This is a simplistic approach, and could be improved for faster flight, for example by sampling along the predicted trajectory of the UAV at high speeds.~~ A potential new sphere's radius is determined in the same way as for the old spheres in the previous step in Eq. 2 with $r_k = 0$. If the potential radius is larger than $r_{min}$, the sphere is added to the map.

*5) 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 our previous work [6]. If any sphere is found to be redundant, it is deleted from the map. ~~This way,~~ large open areas ~~are covered~~ by only a few spheres ~~and can be planned over quickly,~~ and tight corridors ~~have~~ a higher density of spheres, capturing the information about potential paths and distances to obstacles in more detail.

## III. Safe Monocular-Inertial Exploration using the Sphere-Based Map

In this section, we detail how ~~our~~ proposed mapping method and representation are highly suitable for autonomous path planning and exploration, and how volumetric exploration can be achieved, even with only a single camera. In principle, we employ the commonly used receding-horizon next-best-view (RHNBV) [16] strategy, but with several major differences in methodology that are neccessary in the case of a robot with a monocular camera for depth sensing in large-scale, real-world environments.

### A. Frontier Sampling on Free-Space Polyhedron

Since the proposed map representation is fundamentally different from an occupancy grid, we compute the frontiers in a different way than with an occupancy grid — ~~we~~ sample points along the visible free-space polyhedron described in section II 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 some user-defined distance from all map obstacle points. If they do not meet these criteria in any following update, they are deleted.

In 3D outdoor exploration, it is also important to assign different value to different frontiers. We constrain the exploration to only consider frontiers that lie near some obstacle points. Thus, the UAV does not explore up into the sky, and explores near texture-rich areas, ~~leading to fewer possible failure situations due to~~ losing visual odometry tracking. We then add new exploration viewpoints into the map only if a sufficient number of such frontier points would be visible from a given viewpoint at a close distance.

### B. Forced Translation when reaching Viewpoints

In traditional exploration approaches with dense distance sensors, it is often sufficient to move the robot to a boundary between free space and unknown space (known as a frontier), 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 well with a robot ~~with~~ a monocular camera that estimates depth from motion. Such a robot requires *translational* motion to gain correct distance measurements of visual keypoints. With rotation

only, or motions that have too much rotation compared to translation, reliable depth estimates cannot be obtained.

To solve this, when planning a path to a goal viewpoint that could uncover some frontiers, we compute the target headings on the path so that the UAV maintains a fixed heading in the viewpoint's direction for at least $d_c$ meters on the path before reaching the viewpoint, as visualized in Fig. TODO-FIG. This way, the UAV has enough translational motion before reaching a given viewpoint. Therefore, if there are visible visual keypoints from that viewpoint, there will most likely be enough parallax to triangulate these points.

### ~~C. Explored Viewpoint Blocking~~

~~Another necessary distinction is that the UAV blocks 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). Because in our mapping approach, obstacle points correspond to triangulated visual keypoints, no points will be added on textureless surfaces, but they will be on the boundary of free space, so frontiers will be generated there. However, such frontiers cannot be uncovered, since there is nothing behind them, and for this reason, we block the sampling of viewpoints near visited ones, so that the UAV doesn't keep coming back to look at a blank wall.~~

### D. Local-Global Exploration

To allow exploration ~~even~~ in large-scale environments, we employ a local-global approach as in [2] and [20]. The UAV first tries finding paths to any exploration viewpoints up to a user-defined "local exploration radius". If no reachable viewpoints are found in the radius, the UAV tries finding paths to all exploration viewpoints stored in the map. ~~Thanks to the planning efficiency of the sphere graph, this global replanning usually does not take more than a few seconds, but could be made even faster by implementing any sort of long-distance planning abstraction graph, as in [5], [6].~~

### E. Safety-Aware Planning

As a last safety measure, the UAV ~~always flies~~ in the direction of its camera, when not aligning itself with a viewpoint as described in subsection III-B. ~~Free space might always be wrongly initialized, but when flying in the direction of the camera, and with quick periodic replanning, the risk of collision can be minimized.~~ The UAV also plans path with a strong preference for high distance from obstacles, for which the graph-of-spheres representation is ideal, as detailed in [6].

With all of these considerations, a UAV with only a monocular camera and IMU can perform 3D volumetric exploration in large-scale outdoor environments, as we demonstrate experimentally in Sec. section IV. ~~As future work for increasing the safety even more~~, the measurement distance of free-space spheres ~~could be stored and used. With this information, trajectory generation could, for example, force the UAV to~~ fly slower through spheres that have been observed from a large distance and might contain some obstacles not observable from far away.

## IV. Experiments

In this section, we present the performed experiments and show the performance of ~~our~~ method in large-scale real-world (Sec. IV-A) and simulated (Sec. IV-B) environments. Our implementation of the proposed mapping and exploration methods is single-threaded and written in python. As shown in Sec IV-C, even ~~with this simple~~ implementation, the mapping keeps below $1s$ per map update on a standard CPU, and is thus suitable for real-world deployment.

The UAV used in the experiments was equipped with a monocular fisheye Bluefox TODO camera, and TODO IMU. The MRS UAV system [21] was used for trajectory generation and control. In real-world experiments, the camera and IMU are separated from the rest of the UAV by 3D-printed damping elements, which significantly reduces the IMU noise caused by propellers. For state estimation and as the source of sparse visual keypoints, we used OpenVINS [22] in the inverse-depth measurement mode, without loop closures.

### A. Real-World UAV Monocular-Inertial Exploration

~~The most notable~~ real-world experiment is illustrated in Figure 3. ~~In this experiment, the UAV explored up to $60\,\mathrm{m}$ forward around the side of an abandoned farmhouse fully autonomously in an $8$min mission.~~ In this experiment, we bounded the area for generating exploration goals to a $70\,\mathrm{m} \times 10\,\mathrm{m} \times 8\,\mathrm{m}$ bounding box, and the UAV 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 in Figure 3 that the UAV explored 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 the absence of triangulated SLAM keypoints, described in subsubsection II-B.1.

We have tested the proposed approach extensively in a variety of other real-world environments, including a forest path, a tree in a wide open field, and around a large house. Videos from ~~all~~ real-world experiments are available at TODO-YOUTUBE-LINK. ~~Some of these experiments required the safety pilot to take control due to an important limitation of the method that we encountered during testing — since the visual SLAM tracks and triangulates visual key*points*, visual *lines* (caused by e.g. thin tree branches or smooth manmade poles) are not sensed in our mapping pipeline. This caused the UAV to nearly crash into tree crowns multiple times. This limitation could be resolved by integrating short range depth sensors as well, even ultrasound sensors could potentially be fused into the map. Additionally, the visual SLAM could be modified to estimate 3D poses of lines as well, as for example in TODO-CITE, but we leave this for future work.~~

### B. Large-Scale Exploration in Simulation

In this experiment, we showcase the capabilities of our approach in a simulated $120\,\mathrm{m} \times 120\,\mathrm{m} \times 20\,\mathrm{m}$ urban area.


Fig. 3: Visualization of the real-world autonomous exploration experiment described in subsection IV-A, along with a satellite image of the explored abandoned farm area.

The experiment was conducted in the Gazebo simulator, with essentially the same UAV and sensory setup as in the real-world experiments. Figure 4 shows the resulting map of an example exploration mission. ~~After running the exploration mission 20 times, t~~he UAV managed to explore the entire environment in 18/20 runs. In the ~~remaining~~ 2 runs, the UAV crashed either into a large featureless metal pole or the roof of a gazebo where the texture only consists of lines. The crashes were caused by the same limitation as discussed in subsection IV-A. Aside from this limitation, the mapping and exploration can be said to work robustly in static environments. ~~Note also that in the experiment shown in Figure 4, the UAV explored up on the roofs of the buildings when it was able to use the large towers to estimate depth and create enough free space for safely moving up to the roof.~~

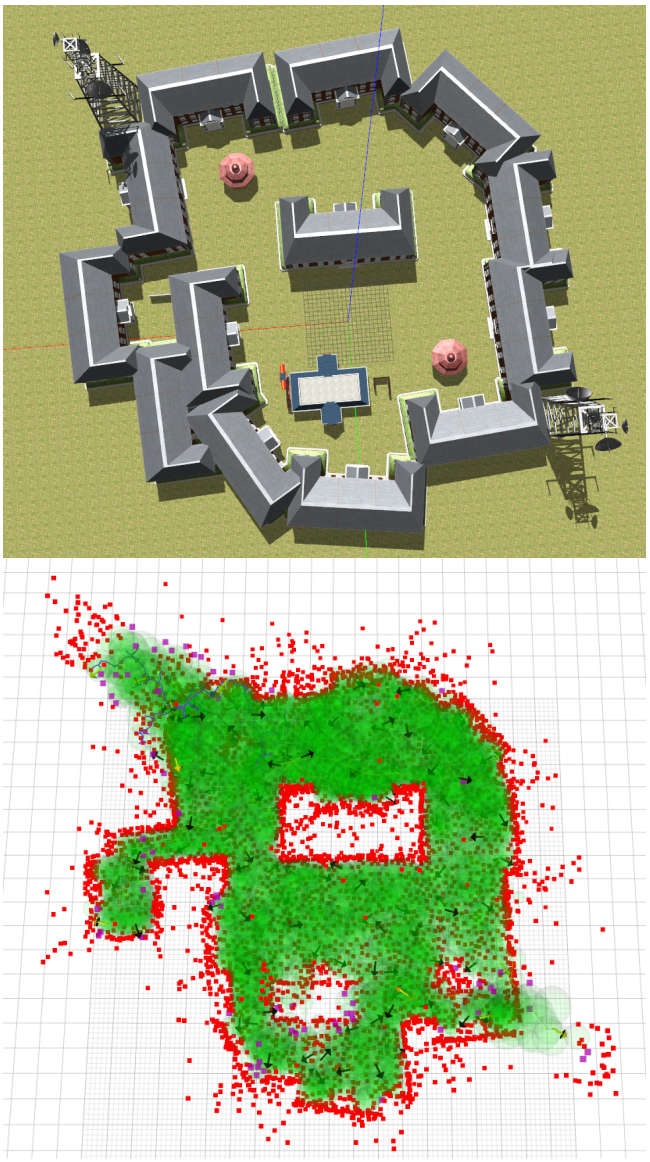

Fig. 4: The resulting map (bottom) after 20min of exploring an urban environment (top) in simulation, explained in more detail in subsection IV-B. Black arrows indicate visited frontier viewpoints, frontiers are shown as purple points. Notice that the UAV also went to explore up on the rooftops, where it could see the metal towers.

### C. Runtime analysis

To provide more insight into the real-time ~~capability~~ of the proposed mapping method, we measured the runtimes of individual parts of the map update iterations during one of the simulated exploration missions described in subsection IV-B and visualized in Figure 4. The resulting graph is shown in Figure 5. In that mission, the UAV explored the entire available environment. Thanks to only updating the map near the UAV, the update time remains relatively bounded, even when the visible free-space polyhedron is large.

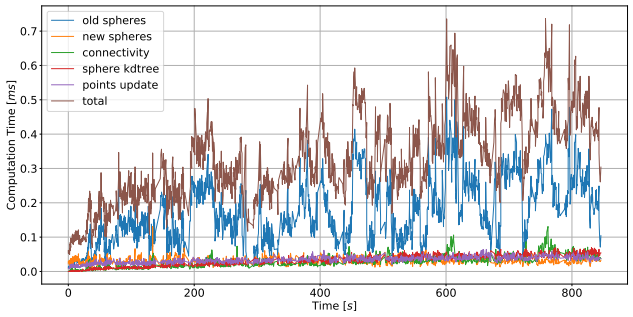


Fig. 5: Runtime analysis of parts of the map update detailed in section II. TODO-MEANINGS OF PARTS

## V. CONCLUSION

The spatial representation proposed in this paper, consisting of spheres and points, offers an alternative to the traditionally used grid-based representations. We have presented a method of constructing this representation on-board a UAV, using only the keypoints from sparse monocular-inertial SLAM as input depth information. We have demonstrated how this new mapping approach enables 3D exploration and navigation in large-scale real-world outdoor environments on a UAV equipped with only a monocular camera and IMU as its sensors.

~~In future work, we intend to explore the potential of the proposed representation for multimodal sensing – e.g. intelligently fusing points and spheres based on the modality that they were created from.~~

## REFERENCES


[1] M. Tranzatto *et al.*, "Team CERBERUS Wins the DARPA Subterranean Challenge: Technical Overview and Lessons Learned," *ArXiv*, vol. abs/2207.04914, 2022.

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

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

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

[5] F. Blöchliger *et al.*, "Topomap: Topological Mapping and Navigation Based on Visual SLAM Maps," *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–9, 2017.

[6] T. Musil *et al.*, "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.

[7] J. Civera *et al.*, "Inverse Depth Parametrization for Monocular SLAM," *IEEE Transactions on Robotics*, vol. 24, pp. 932–945, 2008.

[8] H. Oleynikova *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.

[9] ——, "Sparse 3D Topological Graphs for Micro-Aerial Vehicle Planning," *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1–9, 2018.

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

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

[12] M. Y. Arafat *et al.*, "Vision-Based Navigation Techniques for Unmanned Aerial Vehicles: Review and Challenges," *Drones*, vol. 7, no. 2, 2023.

[13] L. Heng *et al.*, "Autonomous Visual Mapping and Exploration With a Micro Aerial Vehicle," *Journal of Field Robotics*, vol. 31, 2014.

[14] A. Steenbeek *et al.*, "CNN-Based Dense Monocular Visual SLAM for Real-Time UAV Exploration in Emergency Conditions," *Drones*, 2022.

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

[16] 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.

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

[18] A. Bhoi, "Monocular Depth Estimation: A Survey," *ArXiv*, vol. abs/1901.09402, 2019.

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

[20] T. Dang *et al.*, "Graph-based subterranean exploration path planning using aerial and legged robots," *Journal of Field Robotics*, vol. 37, no. 8, pp. 1363–1388, 2020, wiley Online Library.

[21] M. Petrlík *et al.*, "A Robust UAV System for Operations in a Constrained Environment," *IEEE Robotics and Automation Letters (RAL)*, vol. 5, no. 2, pp. 2169–2176, 2020.

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