From sparse SLAM to dense mapping for UAV autonomous navigation

Yassine Habib, Panagiotis Papadakis, Antoine Fagette, Cédric Le Barz, Tiago Gonçalves, Cédric Buche

From sparse SLAM to dense mapping for UAV autonomous navigation

Yassine Habib[a,b], Panagiotis Papadakis[b], Antoine Fagette[c], Cédric Le Barz[a], Tiago Gonçalves[a], and Cédric Buche[d]

[a]Thales, ThereSIS Lab, Palaiseau, France
[b]IMT Atlantique, Lab-STICC, UMR 6285, team RAMBO, Brest, France
[c]Thales, Thales Research & Technology Canada, Montreal, Canada
[d]ENIB, IRL CNRS CROSSING, Adelaide, Australia

ABSTRACT

Autonomous or semi-autonomous navigation of UAVs is of great interest in the Defense and Security domains, as it significantly improves their efficiency and responsiveness during operations. The perception of the environment and in particular the dense and metric 3D mapping in real time is a priority for navigation and obstacle avoidance. We therefore present our strategy to jointly estimate a dense 3D map by combining a sparse map estimated by a state-of-the-art Simultaneous Localization and Mapping (SLAM) system and a dense depth map predicted by a monocular self-supervised method. Then, a lightweight and volumetric multi-view fusion solution is used to build and update a voxel map.

1. INTRODUCTION

In the context of Defense and Security applications, autonomous robot systems have gained considerable interest. In particular, UAVs have provided first responders with a versatile, fast and easily deployable solution to collect information and get a situational awareness before their intervention with most applications requiring an operator to be fully dedicated to piloting. In this case, autonomous or semi-autonomous navigation would allow the operator to control multiple drones or to focus on scene analysis.

A typical use case is the localization of a robot in GNSS-denied environments, as this kind of geo-locatization system is not sufficiently reliable indoors, underground or in warfare. In fact, satellite reception cannot be ensured everywhere, the signal can be spoofed, and its granularity is not adapted for indoor navigation and obstacle avoidance. Other wireless-based systems give a higher precision, but they also involve a strong external dependency for the robot. Specifically in military scenarios, an autonomous system capable of operating even in the absence of external sources is preferred and sometimes required. On the other hand, 3D mapping is a very important function for rescue operations. Autonomous exploration and real-time mapping of a collapsed building is crucial for the intervention of first responders. In this scenario, the structure of the surrounding environment is a priori unknown and irregular, and identifying hazards or locating victims is critical before sending rescuers. It allows to efficiently schedule the intervention and plan the optimal itineraries. To perform such operational reconnaissance, flight assistance or autonomous navigation is essential. It allows the operator to operate the drone in difficult lighting conditions, narrow paths, avoid obstacles, and focus on inspecting visual images.

We restrict our work to the use of an indoor drone, navigating at a reasonable speed for in-building exploration. Therefore, the payload has to meet the Size, Weights and Power (SWaP) constraints. Among a large choice of sensor sets, a calibrated monocular camera synchronized with an Inertial Measurement Unit (IMU) represents one of the most effective solutions. Depth is not observable with monocular cameras but they are cheap, easy to calibrate and widely available. While active sensors such as LiDAR and RGB-D cameras can measure depth, they are more expensive, consume more energy and do not meet the discretionary requirements for certain military uses. As for stereo cameras are concerned, their potential may be limited due to a small baseline distance imposed by the small size of the drone.

Further author information: (Send correspondence to Y. HABIB)
Y. HABIB: E-mail: yassine.habib@thalesgroup.com

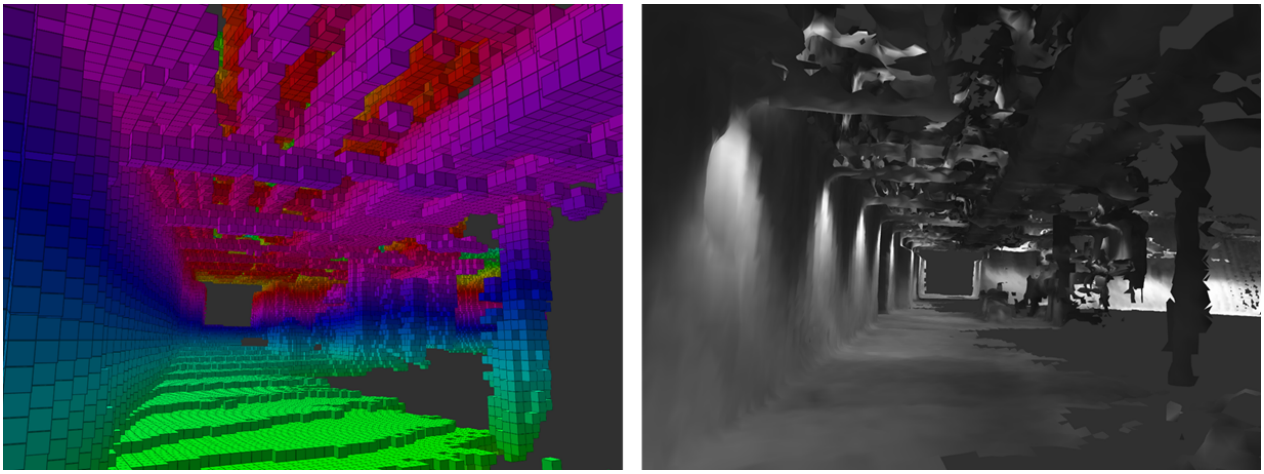

Figure 1. Multi-view mapping by Voxblox[1] on the scene Basement_1 of HILTI SLAM Challenge Dataset.[2] The input data is a set of point clouds measured by a LiDAR sensor. The image on the left shows the constructed voxel map, and the image on the right is the textured 3D mesh deduced after application of marching cubes.

Following a classic drone architecture,[3] an autonomous navigation system requires at least three key functionalities: localization, global environment mapping, and path planning. Here, we will only study the two first tasks. They are typically addressed by Visual SLAM, which aims at mapping the environment from sensor measurements, and localizing the robot in this map in real-time. Although state-of-the-art methods are quite accurate today, they sacrifice map density for computational efficiency by representing the map as a point cloud. In our context, a coarse reconstruction such as a voxel map is more optimal as long as it provides metric and dense structure. An example of voxel map is illustrated on the left image of the Figure 1. Since conventional dense SLAM is overly computationally demanding for real-time operations, a common approach is to leverage Monocular Depth Prediction (MDP) to reconstruct a dense map.

In this paper, we propose a 3D metric, dense and real-time mapping pipeline for UAV navigation. We build it on top of a state-of-the-art SLAM algorithm estimating camera pose and sparse metric map in real-time. A deep learning-based method is then used to predict dense depth from an RGB image and combined with SLAM outputs to densify the estimated map. The dense and metric depth information is then projected onto a voxel map. We design this pipeline for embedded computing on a small drone, and we take this into account in our approach choices. The remainder of this paper is organized as follows. In Section 2, we first review related works in visual SLAM and Monocular Depth Prediction. Then, we present our approach in Section 3. We first make an overview of our navigation system, and then study the robustness of the selected SLAM method. Subsequently, we describe our mapping procedure combining a sparse and metric point cloud with a dense depth map. Lastly, we present the strategy adopted for voxel mapping and multi-view fusion.

2. RELATED WORKS

During past decades, research has focused on optimizing SLAM for real-time localization, achieving top accuracy on classical benchmarks. The main idea is to limit the complexity by keeping the number of map points as small as possible but sufficient for good robustness and precision. Monocular-inertial methods are quite advanced, enabling precise localization and sparse 3D metric maps. However, obstacle avoidance based on space-occupancy requires a dense representation which is still not achievable in real-time with monocular-inertial configuration. Yet, since the Monocular Depth Prediction has benefited from advances in deep learning, it can help predict depth maps to provide a dense representation.

2.1 Conventional Visual SLAM

Dense mapping by SLAM is usually achieved by direct methods[4] that estimate camera pose by image alignment minimizing the photometric error. While providing more robustness in bad visual conditions or pure rotations,

they have a high computational cost. Instead, semi-direct approaches[5,6] use patches around tracked keypoints instead of whole image pixels, combining photometric and reprojection error. Kimera[7] adapts SVO+GTSAM[6] for stereo and extends it to perform real-time, dense, metric and semantic SLAM. A sparse map is built and maintained for localization. In parallel, a dense depth map is estimated by stereo matching and projected using the estimated camera pose with Voxblox.[1] Then, a 3D mesh is extracted from the voxel map by marching cubes.

Finally, feature-based methods give the best efficiency since they only rely on a sparse set of keypoints. Since the number of points is limited, global Bundle Adjustment (BA) can be considered to jointly optimize camera poses and landmarks (map points). ORB-SLAM[8] proposed a pure monocular algorithm running in real-time using a more robust version of the ORB feature descriptor, a covisibility graph representation, and loop closure detection. This work was extended for multiple sensors configurations[9,10] and became ORB-SLAM 3.[11] It is state of the art and implements relevant features for robotic applications such as the Atlas multi-map system or being camera model agnostic. However, even though the IMU initialization process has been accelerated, it is still suboptimal. In slow motion, the initialization tends to fail, especially without pitch and roll rotations. On the other hand, Basalt[12] leverages Double Sphere camera model and adopts a fixed-lag strategy with a nonlinear factor recovery procedure. The tracking thread is restrained to local BA on a window. Each time a keyframe is marginalized out of the optimization window, ORB features are extracted and non linear factors (roll-pitch, yaw, absolute position and relative poses) are recovered before global BA is performed. We decide to build our approach on top of ORB-SLAM 3[11] since it is the state of the art among monocular inertial SLAM and works in real time.

2.2 Deep Learning in SLAM

Recently, Deep Learning was applied for SLAM by end-to-end learning or for specific tasks. Most of these methods do not outperform conventional methods for localization. Yet, Droid-SLAM[13] recently achieved competitive results with a better robustness and building a dense map, but it relies on a massive Deep Neural Network (DNN) not adapted for embedded computing. On the other hand, in order to produce a dense 3D map, some authors proposed to densify existing semi-dense SLAM solutions using MDP. CNN-SLAM[14] predicts a depth map from a monocular image using a Convolutional Neural Network (CNN) and fuses it with depth measurements of LSD-SLAM.[5] CodeSLAM[15] introduced the use of Variational AutoEncoders (VAE) to infer depth, learning a compact depth representation from an RGB image and a SLAM-based sparse map. DeepFactors[16] and CodeMapping[17] implement the CodeSLAM VAE in a full SLAM system and further add depth uncertainty prediction and multi-view refinement via a factor graph. The latter is built on ORB-SLAM 3[11] and leverages available reprojection error to account for sparse depth uncertainty. It shows promising results, but the DNN is heavy and trained in a supervised fashion leading to limited generalization capacity. We use this type of approach while trying to minimize the complexity, and use self-supervised learning.

2.3 Monocular Depth Prediction

The main challenge of using monocular cameras for SLAM is that depth is not observable. The idea behind Monocular Depth Prediction is that we can recover depth from semantics and observed structures in the image. In this topic, deep learning has brought considerable improvements over geometric methods. Eigen[18] introduced the topic with a multi-scale CNN architecture, combining a coarse network predicting global depth and finer one for local depth. The conventional evaluation metrics and procedure were also defined. Training these networks requires a large amount of data with a lot of variability and depth ground truth. For this reason, Garg et al.[19] presented a self-supervised training procedure based on multi-view geometry. Using stereo pairs, the idea is to infer a depth map from the left image, and try to reconstruct it using right image and the predicted depth. The reconstruction error to minimize is the photometric error, where high image resolution with more texture details improves performances. This is extended to monocular sequences as in[20–22] where a dedicated network jointly learns to infer the relative pose between images. With this approach, Packnet-Sfm[23] implements packing and unpacking blocks through 3D convolutions to avoid pooling operations. This is supposed to preserve the dense geometric and appearance details as long as possible in the network. The model achieves competitive results similar to fully-supervised models. Therefore, we decided to combine the ORB-SLAM 3[11] with Packnet-Sfm,[23] as both methods seem to perform well in their respective tasks. We believe that a simple but effective fusion strategy would be sufficient to produce dense and metric depths from the results of these algorithms.

3. PROPOSED APPROACH

In the context of our study, we design a system to produce a dense map for UAV autonomous navigation. We base our work on a state-of-the-art monocular-inertial SLAM, and densify the estimated sparse map by leveraging Deep Learning-based Monocular Depth Prediction.

3.1 Navigation system overview

We seek to design a navigation system that can be embedded on a drone for real-time operation. To this end, the configuration of the architecture is essential. As the perception of the environment involves multiple sensors and extensive computation, every module in the data pipeline should be optimized. Nowadays, many embedded systems typically include hardware accelerators dedicated to camera acquisition, and thus minimizing CPU usage. Furthermore, since most SLAM methods are CPU-based, this leaves room within the GPU for running deep learning-based MDP solutions. A relevant example is the NVIDIA Jetson series of embedded systems.

In the context of our study, we choose to base our work on a state-of-the-art sparse SLAM algorithm. The criterion here is to choose an accurate solution able to run in real-time. Nowadays, most methods adopt a multi-threaded architecture. As represented in Figure 2, a Tracking thread runs at camera frame rate to estimate camera pose from a monocular image and pre-integrated inertial measurements. At this stage, a sparse set of keypoints is usually extracted and tracked over a set of successive frames for pose estimation. Some of these frames matching specific conditions, including an overlapping threshold, are selected to become keyframes. Then, the Local Mapping processes each keyframe to triangulate keypoints and update the local map at keyframe rate. Finally, some methods implement Global Mapping in a dedicated thread, performing place recognition and loop closure at a lower frequency. This task can be very time consuming, so a sparse map representation is always preferred for this operation.

For in-building exploration at a reasonable flight speed, we can argue from our experiments that the average Keyframe rate remains low, between 2 and 4 keyframes per second on EuRoC[24] and UZH-FPV[25] datasets, and around 6 keyframes per second on HILTI SLAM Challenge Dataset.[2] As suggested in[26] where SLAM algorithms are benchmarked on several NVIDIA Jetson platforms, sparse SLAM has a significant load on the CPU but leaves the GPU completely available for depth prediction. This means that we can consider using a DNN to predict a dense depth map for each keyframe. Furthermore, since MDP cannot ensure a metric scale, we suggest to fuse the predicted dense depth map with the sparse metric depth estimated by SLAM in the Mapping thread. Subsequently, inspired by Kimera,[7] we use Voxblox[1] to build a voxel map from the scaled depth map and the estimated camera pose. The algorithm builds a Truncated Signed Distance Function (TSDF) by bundled

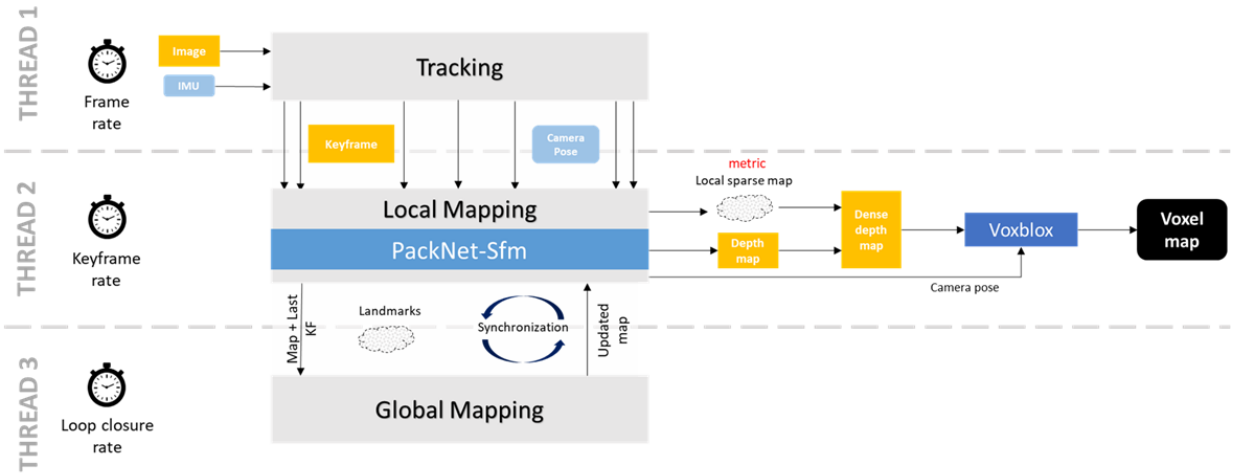

Figure 2. Scheme of our proposed pipeline based on multi-threaded SLAM. The Mapping thread is extended with dense depth map prediction. The predicted dense depth and metric sparse depth are then fused together. The keyframe pose along with the metric dense depth map are processed by Voxblox[1] to build and update the voxel map.

Table 1. RMS ATE (meters) of Kimera,[7] ORB-SLAM 3[11] and Basalt[12] on EuRoC[24] dataset.

Method	MH01	MH02	MH03	MH04	MH05	V101	V102	V103	V201	V202	V203	Avg
ORB-SLAM 3 (MI)	**0.031**	0.053	0.034	0.129	**0.073**	0.044	**0.015**	**0.022**	0.039	0.019	**0.022**	0.044
Basalt	0.082	0.041	0.050	0.095	0.133	0.043	0.049	0.060	0.039	0.053	0.239	0.080
Kimera	0.112	0.090	0.079	3.533	0.173	0.060	0.059	0.175	0.051	0.081	0.394	0.437
ORB-SLAM 3	0.037	**0.028**	**0.027**	**0.052**	0.083	**0.036**	**0.015**	0.025	**0.026**	**0.014**	0.026	**0.034**

raycasting, which means that pixels that fall on the same voxel are raycasted together. This approach is optimal and runs only on CPU, but could also be ported to GPU using nvblox.[27]

3.2 Sparse SLAM baseline

In 3D SLAM, the trajectory estimation is a crucial part because landmarks are usually projected from estimated camera poses. However, keeping track of the camera ego-motion in challenging scenes is difficult.We chose ORB-SLAM 3[11] as a baseline SLAM method. In this section, we confirm this choice by comparing it to other competitive methods, Basalt and Kimera,[7] in different UAVs scenarios including indoor navigation, fast motions and illumination changes.

Most of these SLAM methods were already benchmarked on EuRoC[24] dataset that was generated by a Micro-Aerial Vehicle (MAV) in 2 different small rooms, so it does not contain large environments and very few challenging sensing conditions. On the other hand, the UZH FPV Drone Racing dataset[25] (UZH-FPV) provides stereo inertial data acquired from an FPV drone which obviously contains fast motions, in indoor and outdoor environments. Finally, the HILTI SLAM Challenge Dataset[2] released in 2021 was meant to benchmark SLAM algorithms on large indoor and outdoor environments, including low textures, illumination changes and transparent surfaces. It was collected by a hand-held payload including calibrated and synchronized multi-sensors.

We first evaluate the algorithms on EuRoC[24] trying to reproduce the reported results and ensure that we correctly setup each algorithm. We used evo[28] library to measure the Root Mean Square (RMS) of the Absolute Trajectory Error (ATE) on SE(3) as done by the authors. We run each method 10 times in each scene and report the median. Results are shown in Table 1 and they globally correspond to the reported results from Basalt[12] and ORB-SLAM 3[11] with an average difference of 0.03m. However, we highlight that we could not reproduce the same results for Kimera[7] for which we observe an average difference of 0.30m, especially in difficult scenes. Here, we do not analyze further these results as their authors already did, but we globally note good robustness and precision, except for Kimera[7] on difficult scenes.

In the sequel, we benchmark these algorithms on the UZH-FPV dataset[25] to challenge them under high dynamic motions. We do not report the results for the scenes captured with the camera facing down, but only those with the camera facing forward which is more in line with the building exploration scenario. Basalt[12] was already configured for this dataset by their authors. In fact, it shows a high robustness with a negligible standard deviation (std) on the 10 runs. As we can see in Figure 3, the trajectory shape appears to be accurate, but the scale error decreases the accuracy. Results in Table 2 show that ORB-SLAM 3[11] in Monocular Inertial (MI)

Table 2. RMS ATE (meters) of Kimera,[7] ORB-SLAM 3[11] and Basalt[12] on UZH-FPV dataset.[25]

	indoor_forward						outdoor_forward			Avg
	10	3	5	6	7	9	1	3	5	
ORB-SLAM 3 (MI)	**0.607**	0.508	**1.041**	**0.432**	**0.167**	**0.575**	**1.200**	**1.051**	X	0.698
Basalt	1.520	0.731	1.130	1.404	1.268	1.583	1.618	2.126	**3.031**	1.601
Kimera	3.502	3.365	2.998	6.017	6.251	3.533	X	14.367	11.269	6.413
ORB-SLAM 3	0.877	**0.493**	1.368	0.505	0.341	0.953	1.383	1.553	X	0.934

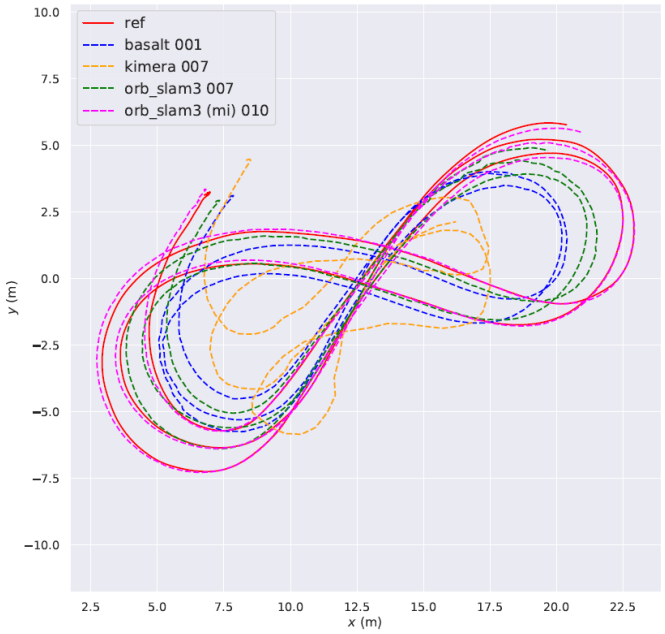

Figure 3. Trajectories estimated by ORB-SLAM 3,[11] Basalt[12] and Kimera[7] on UZH-FPV[25] indoor_forward_9.

configuration has the best results. And yet, it also suffers from scale error and is less stable with an average std of 0.373m and up to 2.05m on the most difficult scene. The indoor scenes top speed is 12.8 m/s and the mean error range from 0.167m to 1.041m which is not so bad considering the high speed and the important optical flow. Regarding Kimera,[7] it appears to have great difficulty in keeping a good track of the drone trajectory, especially when estimating rotations in turns. Perhaps a thorough tuning of the algorithm's parameters would improve the results.

The HILTI SLAM Challenge Dataset[2] provides very interesting scenes for our use case such as long trajectories inside a building with illumination changes. However, they do not provide full trajectory ground truth for each scene, but only intermittent poses. Also, calibration data were not published, so we are not able to use the Double Sphere camera model preferred for Basalt.[12] The scene Basement_1 is particularly interesting because it is composed of underground corridors whose lights turn on when the operator approaches them, which induces strong variations of lighting. The preliminary results using ORB-SLAM 3[11] are presented on Figure 4. The algorithm is capable of estimating accurate trajectories, with a RMSE ranging from 0.02m to 0.35m.

These first experiments suggest that ORB-SLAM 3[11] in monocular-inertial configuration is a relevant SLAM baseline for our drone navigation system. Additionally, qualitative results on HILTI SLAM Challenge Dataset[2] are promising since the algorithm manages to perform tracking despite strong illumination changes in underground corridors. An initialization delay is also observed at the beginning of some scenes where there is not enough motion for ORB-SLAM 3[11] to initialize the IMU.

3.3 Map densification and scaling

Although ORB-SLAM 3[11] can provide accurate localization, it estimates a sparse map as a point cloud. However, each landmark is the result of Bundle Adjustment on the covisibility graph, which means that they include information from multiple views. Moreover, the corresponding reprojection error provides a measure of uncertainty for each map point. We argue that this sparse depth would allow recovery of the metric scale of a depth map inferred by a DNN, as the one shown in Figure 5.

We choose Packnet-Sfm[23] for depth prediction because it is among the most accurate state-of-the-art self-supervised methods with a publicly available code. It is supposed to be scale consistent, which means that the

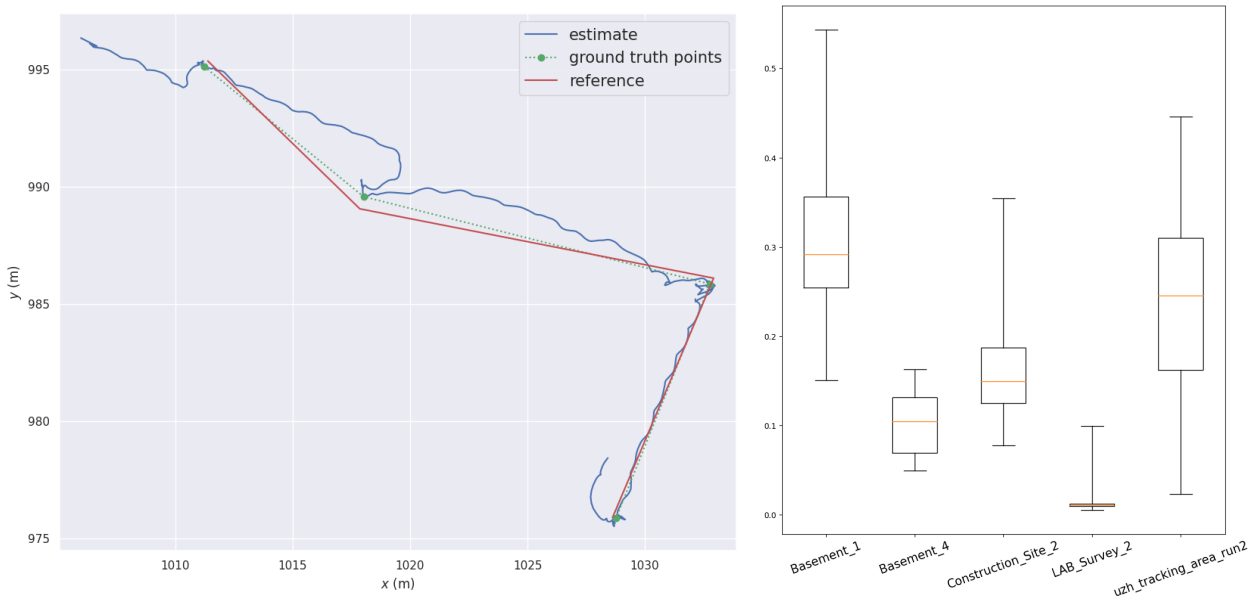

Figure 4. ORB-SLAM 3[11] (Monocular-Inertial)[11] results on HILTI SLAM Challenge Dataset.[2] Left image: Trajectory estimated on scene Basement_1. Right image: Boxplots of the ATE on different scenes.

depth map is predicted up to a global scale factor α. The scale is usually extracted from sensor depth measurements by calculating the ratio of median depths between ground truth and predicted depths.[20] Nevertheless, the ground truth is usually collected from a LiDAR sensor. Instead, as presented in,[30] we use a loosely-coupled approach to recover scale by minimizing the square relative error between ORB-SLAM 3[11] sparse depth points and corresponding inferred depth.

$$\hat{\alpha} = \min_{\alpha} \frac{1}{N} \sum_{p \in \Omega} \frac{\left\| \alpha \hat{D}_p - D_p \right\|^2}{D_p} \tag{1}$$

where Ω is the set of pixels p tracked by SLAM, N is the number of points in Ω, D_p is the triangulated depth and $\hat{D}_p$ is the depth predicted by the neural network. The scene V1_01 of EuRoC[24] dataset provides indoor data suitable for SLAM and with accurate pose and structure ground truth. Therefore, we could make our experiments by first running ORB-SLAM 3[11] and saving keyframes with their sparse depths. Afterwards, we use Packnet-Sfm[23] to infer a dense depth map for all keyframes. We apply the classical metrics of Monocular Depth Prediction defined in.[18]

Quantitative results were presented in our previous work[30] where we evaluated the predicted depth map against the ground truth without scaling, with ground truth scale, and with SLAM scale (1). We concluded that the estimated sparse depth does indeed allow to retrieve a good approximation of the global scale factor. We justify in more details in Figure 6 where the difference between ground truth and SLAM scale is really small, with a mean absolute difference of 3.2%. We can also observe that the scale of the predicted depth maps are not consistent over time.

However, qualitative results shown in Figure 5 demonstrate that the predictions are not accurate enough on indoor scenes. In fact, we used pre-trained weights provided by the author that were trained on KITTI dataset[31] which consists of outdoor images collected from a car. While the segmentation of object surfaces is correct, the corresponding depth is subject to errors. For instance, the walls' surface normal is not completely perpendicular to the floor. Also, the predicted depths in the upper part of the image are very large. We link this to the training dataset where this specific area contains mostly sky. Fine-tuning the neural network on indoor scenes should minimize this issue.

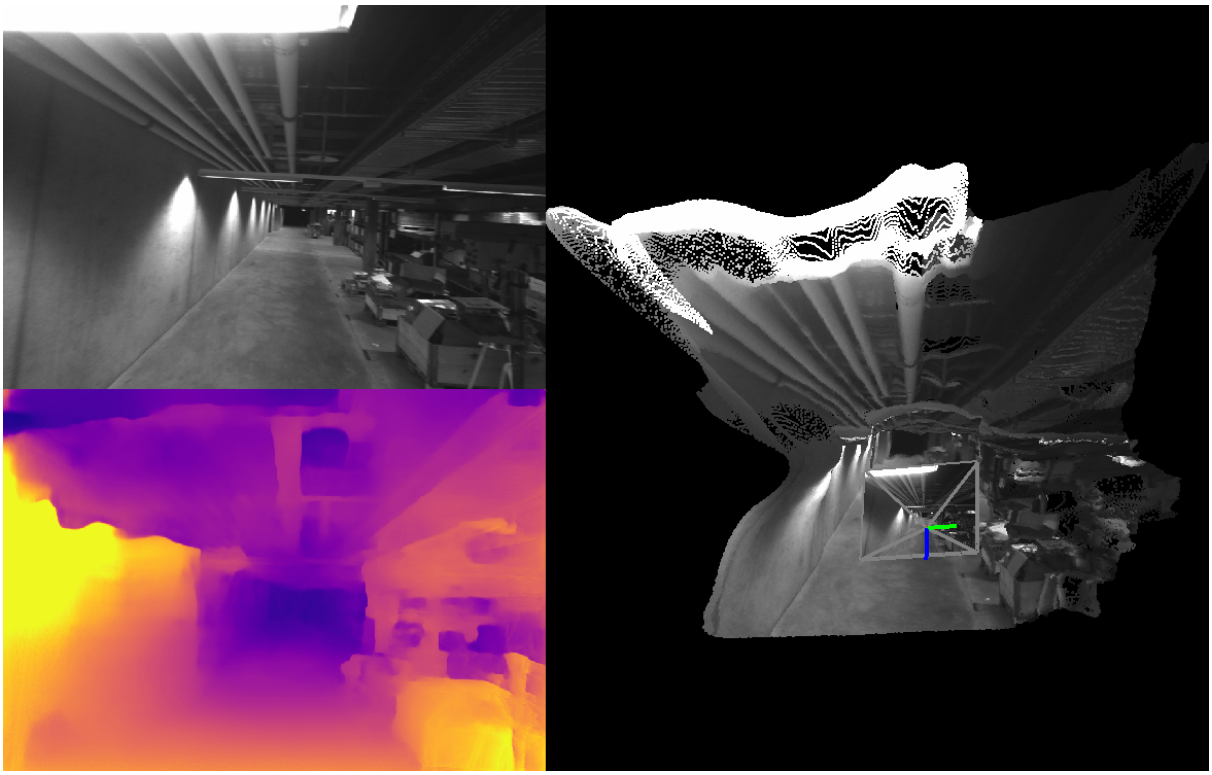

Figure 5. Packnet-Sfm[23] results on HILTI SLAM Challenge Dataset,[2] scene Basement_1. The top-left image is the original monocular image. The bottom-left image is the inferred depth map. The right image is the 3D visualization of the predicted depth map using camviz:[29] depth points are projected in 3D using camera intrinsic parameters.

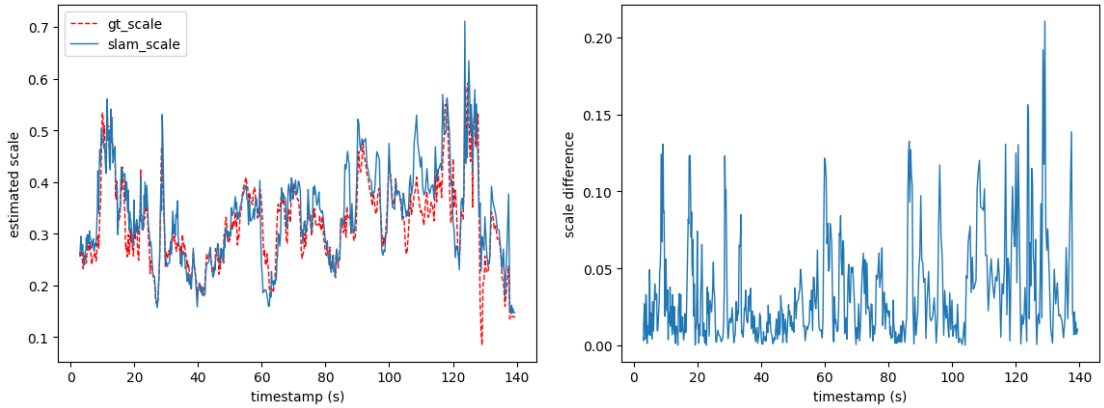

Figure 6. Global scale factors estimated for Packnet-Sfm[23] predictions on the Basement_1 scene of HILTI SLAM Challenge Dataset.[2] The ground truth scale is computed using the median depths and the SLAM scale refers to our approach. The plot on the right is the absolute difference between the 2 approaches, where the mean difference is 3.2%.

In addition, we did not fully exploit the estimated depth, as it could be used to improve depth prediction. While SLAM estimates depth based on epipolar geometry, it is more difficult to formally justify the Neural Network predictions. In fact, SLAM triangulates map points from multiple views and optimize a pose graph by minimizing a reprojection error. A tightly coupled approach would allow to improve the prediction itself and not only recover scale. On the other hand, the reprojection error gives an idea of the confidence of each point and should be used to weigh their contribution.

3.4 Multi-view fusion

Other works densifying SLAM[16,17] leverage multi-view fusion through factor graph to improve the accuracy. Instead, we follow Kimera[7] approach relying on a fast and iterative volumetric method. Thus, provided with a scaled dense depth map, we use Voxblox[1] grouped raycasting to construct a voxel map. Voxels include a weight and a distance to the closest surface. To create and maintain the TSDF, each pixel is raycasted from the image plane to its 3D position. During raycasting, each crossed voxel is updated if it is within the truncation distance. The weighing function is 1 before the surface and drops quadratically behind it. Repeating this procedure from multiple views provides a fast volumetric fusion.

We experiment this method on the scene Basement_1 of HILTI SLAM Challenge Dataset.[2] We use as input the ORB-SLAM 3[11] estimated sparse depths and the Packnet-Sfm[23] depth scaled with our approach. Voxel maps and corresponding 3D meshes are shown in Figure 7. We also show for reference the results on the LiDAR measurements in Figure 1. The sparse depth provides a good insight of the geometric structure but truly lacks of density. Moreover, some outliers remain on the voxel map and typically correspond to points with high reprojection error. Regarding the densified depth, the multi-view fusion is not sufficient to correct the structure error. We could probably distinguish the long corridor but the map is too coarse and irregular to be used for navigation.

On this task, we plan to take into account the depth map uncertainty that could be computed from reprojection error[30] or directly inferred from the neural network. Indeed, Voxblox[1] weighing function should be adapted to account for each depth point uncertainty to avoid corrupting the map with wrong predictions.

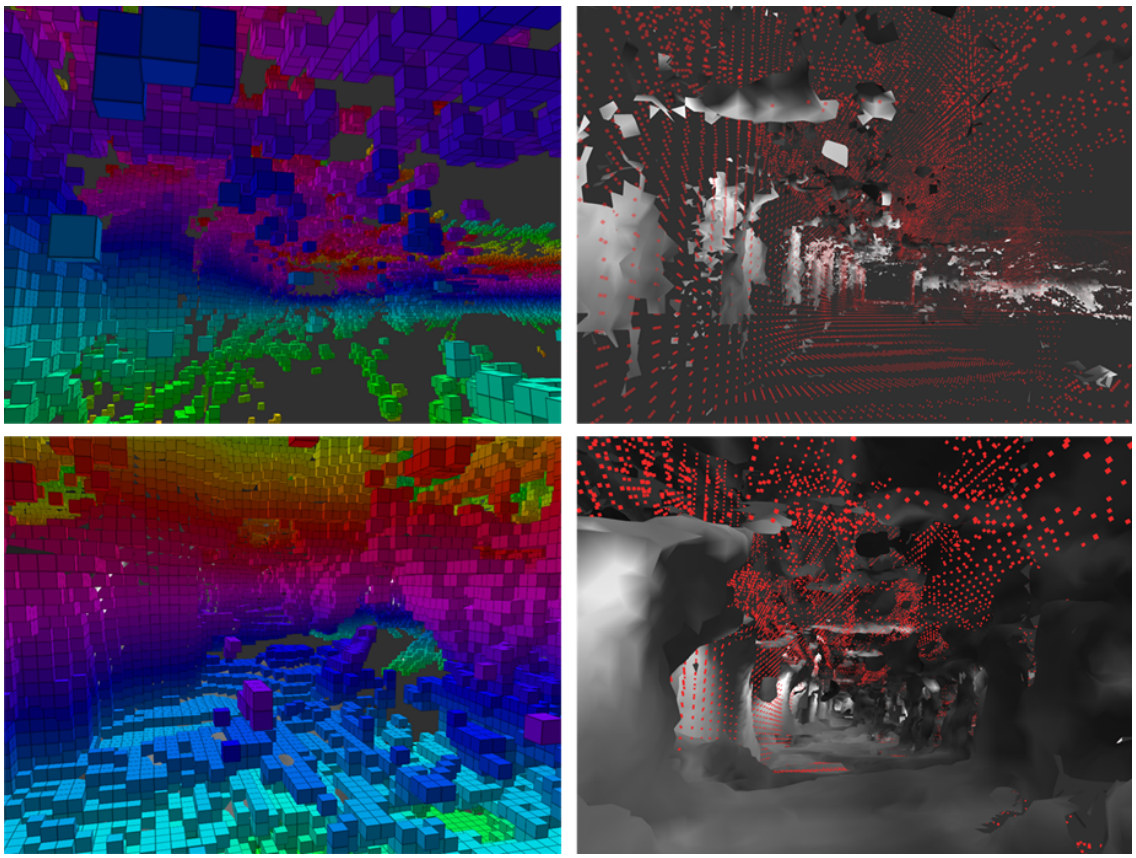

Figure 7. Multi-view volumetric fusion using Voxblox,[1] where the left image shows the voxel map, and the right image the 3D mesh reconstructed with marching cubes. The LiDAR measured point cloud is displayed in red for reference. The first line is the result on ORB-SLAM 3[11] estimated sparse depth. The second line is the predicted depth densified with our approach.

4. CONCLUSION

We have presented in this paper how monocular-inertial SLAM can be used for autonomous indoor navigation in the context of defense, security or rescue applications. Building a dense metric map in real-time is of high interest for UAV autonomous navigation but is too computationally demanding to maintain. In this context, we propose a navigation system architecture that could fit in an embedded system for real-time operations. It combines a state-of-the-art SLAM estimating sparse metric depth, and a MDP network inferring a dense depth map. We chose ORB-SLAM 3[11] as a baseline after benchmarking state-of-the-art SLAM on various drone datasets, including scenes with high motions and illumination changes. Our loosely coupled approach to densify SLAM sparse map is defined. It consists in leveraging Packnet-Sfm[23] to infer a dense depth map and to retrieve metric scale from sparse depth estimated by SLAM. The preliminary results are presented and the current limitations are highlighted.

Our on-going work aims at first improving the predicted depth by employing a tightly coupled approach. By providing the sparse depth triangulated by SLAM to the DNN, we want it to leverage the estimated metric depth to better infer the depth map. We also plan to include the reprojection error, and to employ uncertainty map prediction.

ACKNOWLEDGMENTS

We would like to thank the ANRT (Association Nationale de la Recherche et de la Technologie), for its funding through the CIFRE grant 2019/1877. We must also mention Cédric LE BARZ for his contribution to this work. We appreciated his dedication and support in supervising some of the tasks in this research.

REFERENCES

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

[2] Helmberger, M., Morin, K., Kumar, N., Wang, D., Yue, Y., Cioffi, G., and Scaramuzza, D., "The hilti slam challenge dataset," (2021).

[3] Hanover, D., Loquercio, A., Bauersfeld, L., Romero, A., Penicka, R., Song, Y., Cioffi, G., Kaufmann, E., and Scaramuzza, D., "Autonomous drone racing: A survey," (2023).

[4] Newcombe, R. A., Lovegrove, S. J., and Davison, A. J., "Dtam: Dense tracking and mapping in real-time," in [*2011 International Conference on Computer Vision*], 2320–2327 (2011).

[5] Engel, J., Schöps, T., and Cremers, D., "Lsd-slam: Large-scale direct monocular slam," in [*European conference on computer vision*], 834–849, Springer (2014).

[6] Forster, C., Carlone, L., Dellaert, F., and Scaramuzza, D., "On-manifold preintegration for real-time visual–inertial odometry," *IEEE Transactions on Robotics* **33**(1), 1–21 (2017).

[7] Rosinol, A., Abate, M., Chang, Y., and Carlone, L., "Kimera: an open-source library for real-time metric-semantic localization and mapping," in [*IEEE International Conference on Robotics and Automation (ICRA)*], 1689–1696 (2020).

[8] Mur-Artal, R., Montiel, J. M. M., and Tardós, J. D., "Orb-slam: A versatile and accurate monocular slam system," *IEEE Transactions on Robotics* **31**(5), 1147–1163 (2015).

[9] Mur-Artal, R. and Tardós, J. D., "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE Transactions on Robotics* **33**(5), 1255–1262 (2017).

[10] Mur-Artal, R. and Tardós, J. D., "Visual-inertial monocular slam with map reuse," *IEEE Robotics and Automation Letters* **2**(2), 796–803 (2017).

[11] Campos, C., Elvira, R., Rodríguez, J. J. G., Montiel, J. M. M., and Tardós, J. D., "Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam," *IEEE Transactions on Robotics* **37**(6), 1874–1890 (2021).

[12] Usenko, V., Demmel, N., Schubert, D., Stückler, J., and Cremers, D., "Visual-inertial mapping with non-linear factor recovery," *IEEE Robotics and Automation Letters* **5**(2), 422–429 (2020).

[13] Teed, Z. and Deng, J., "DROID-SLAM: deep visual SLAM for monocular, stereo, and RGB-D cameras," *CoRR* **abs/2108.10869** (2021).

[14] Tateno, K., Tombari, F., Laina, I., and Navab, N., "CNN-SLAM: real-time dense monocular SLAM with learned depth prediction," *CoRR* **abs/1704.03489** (2017).

[15] Bloesch, M., Czarnowski, J., Clark, R., Leutenegger, S., and Davison, A. J., "Codeslam—learning a compact, optimisable representation for dense visual slam," in [*IEEE conference on computer vision and pattern recognition*], 2560–2568 (2018).

[16] Czarnowski, J., Laidlow, T., Clark, R., and Davison, A. J., "Deepfactors: Real-time probabilistic dense monocular slam," *IEEE Robotics and Automation Letters* **5**(2), 721–728 (2020).

[17] Matsuki, H., Scona, R., Czarnowski, J., and Davison, A. J., "Codemapping: Real-time dense mapping for sparse slam using compact scene representations," *IEEE Robotics and Automation Letters* **6**(4), 7105–7112 (2021).

[18] Eigen, D., Puhrsch, C., and Fergus, R., "Depth map prediction from a single image using a multi-scale deep network," *Advances in neural information processing systems* **27**, 2366–2374 (2014).

[19] Garg, R., Bg, V. K., Carneiro, G., and Reid, I., "Unsupervised cnn for single view depth estimation: Geometry to the rescue," in [*European conference on computer vision*], 740–756, Springer (2016).

[20] Zhou, T., Brown, M., Snavely, N., and Lowe, D. G., "Unsupervised learning of depth and ego-motion from video," in [*IEEE conference on computer vision and pattern recognition (CVPR)*], 1851–1858 (2017).

[21] Godard, C., Mac Aodha, O., and Brostow, G. J., "Unsupervised monocular depth estimation with left-right consistency," in [*IEEE conference on computer vision and pattern recognition (CVPR)*], 270–279 (2017).

[22] Godard, C., Aodha, O. M., Firman, M., and Brostow, G., "Digging into self-supervised monocular depth estimation," in [*2019 IEEE/CVF International Conference on Computer Vision (ICCV)*], 3827–3837 (2019).

[23] Guizilini, V., Ambrus, R., Pillai, S., Raventos, A., and Gaidon, A., "3d packing for self-supervised monocular depth estimation," in [*IEEE Conference on Computer Vision and Pattern Recognition*], 2485–2494 (2020).

[24] Burri, M., Nikolic, J., Gohl, P., Schneider, T., Rehder, J., Omari, S., Achtelik, M. W., and Siegwart, R., "The euroc micro aerial vehicle datasets," *The International Journal of Robotics Research (IJRR)* **35**(10), 1157–1163 (2016).

[25] Delmerico, J., Cieslewski, T., Rebecq, H., Faessler, M., and Scaramuzza, D., "Are we ready for autonomous drone racing? the uzh-fpv drone racing dataset," in [*2019 International Conference on Robotics and Automation (ICRA)*], 6713–6719 (2019).

[26] Jeon, J., Jung, S., Lee, E., Choi, D., and Myung, H., "Run your visual-inertial odometry on nvidia jetson: Benchmark tests on a micro aerial vehicle," *IEEE Robotics and Automation Letters* **6**(3), 5332–5339 (2021).

[27] NVIDIA, "nvblox: Signed distance functions (sdfs) on nvidia gpus.." https://github.com/nvidia-isaac/nvblox (2022).

[28] Grupp, M., "evo: Python package for the evaluation of odometry and slam.." https://github.com/MichaelGrupp/evo (2017).

[29] Institute, T. R., "Camviz." https://github.com/TRI-ML/camviz (2021).

[30] Habib, Y., Papadakis, P., Le Barz, C., Fagette, A., Gonçalves, T., and Buche, C., "Densifying slam for uav navigation by fusion of monocular depth prediction," in [*IEEE International Conference on Automation, Robotics and Applications*], (2023).

[31] Geiger, A., Lenz, P., and Urtasun, R., "Are we ready for autonomous driving? the kitti vision benchmark suite," in [*Conference on Computer Vision and Pattern Recognition (CVPR)*], (2012).