Communication

CNN-Based Dense Monocular Visual SLAM for Real-Time UAV Exploration in Emergency Conditions

Anne Steenbeek and Francesco Nex *

Faculty of Geo-Information Science and Earth Observation (ITC), University of Twente, 7514 AE Enschede, The Netherlands; h.a.steenbeek@utwente.nl
* Correspondence: f.nex@utwente.nl

Abstract: Unmanned Aerial Vehicles (UAVs) for 3D indoor mapping applications are often equipped with bulky and expensive sensors, such as LIDAR (Light Detection and Ranging) or depth cameras. The same task could also be performed by inexpensive RGB cameras installed on light and small platforms that are more agile to move in confined spaces, such as during emergencies. However, this task is still challenging because of the absence of a GNSS (Global Navigation Satellite System) signal that limits the localization (and scaling) of the UAV. The reduced density of points in feature-based monocular SLAM (Simultaneous Localization and Mapping) then limits the completeness of the delivered maps. In this paper, the real-time capabilities of a commercial, inexpensive UAV (DJI Tello) for indoor mapping are investigated. The work aims to assess its suitability for quick mapping in emergency conditions to support First Responders (FR) during rescue operations in collapsed buildings. The proposed solution only uses images in input and integrates SLAM and CNN-based (Convolutional Neural Networks) Single Image Depth Estimation (SIDE) algorithms to densify and scale the data and to deliver a map of the environment suitable for real-time exploration. The implemented algorithms, the training strategy of the network, and the first tests on the main elements of the proposed methodology are reported in detail. The results achieved in real indoor environments are also presented, demonstrating performances that are compatible with FRs' requirements to explore indoor volumes before entering the building.

Keywords: UAV; 3D map; indoor; real-time; CNN; Single Image Depth Estimation; emergency

1. Introduction

The quick generation of 3D maps of indoor environments is of primary importance during rescue operations to analyze confined spaces, preventing First Responders (FR) from entering unstable, collapsed buildings and risking their lives. In this regard, Unmanned Aerial Vehicles (UAV) have shown to be a valid solution to quickly generate detailed 3D maps within the last decade. Although a consistent number of UAV mapping solutions has been developed in outdoor environments [1], we have seen a growing number of studies specifically conceived to work in confined indoor spaces [2–4] in recent years. Compared to outdoor space mapping, indoor mapping has some additional challenges [5]. For example, GNSS instruments cannot be used, preventing the possibility to localize the platform and scale the scene, while the restricted spaces require the use of small and agile platforms to move. According to the application, illumination can be limited, making the registration of the images even more difficult [6]. A dense reconstruction of the environment cannot be produced by simply using most of the classical SLAM (Simultaneous Localization and Mapping) algorithms. For these reasons, most of the published research has focused on the use of advanced platforms integrating different sensors [7–9]. Devices such as LiDAR, ultrasound, or stereo-rig cameras allow distances to be measured, while inertial units support the navigation and the scaling of the scene [10]. However, these instruments make platforms bulkier in most cases. A few miniaturized solutions, specifically customized to
work in narrow spaces, have recently shown extremely promising results, but they still are relatively expensive with several customizations [11]. The usability of these solutions is also limited in several practical cases: the use in harsh and risky environments requires cheap and easily replicable solutions to be replaceable in case of damage. On the other hand, there are still no solutions using commercial platforms for real-time indoor mapping applications. This is mainly because most of the available platforms (and SDK, Software Development Kit) do not allow for data to be acquired from the on-board IMU, making RGB video sequences the only information to exploit for mapping purposes.

The surge of deep learning has been influencing the development of algorithms for many UAV tasks. In the literature, algorithms embedding CNNs to autonomously fly a UAV [12–14], to reconstruct a 3D environment using single and stereo images [15–17], and to efficiently segment a scene [18] using UAV images have already been proposed. If real-time processing is a must for autonomous navigation, 3D reconstruction and scene segmentation are mainly performed in post-processing. Real-time algorithms are mostly performed on customized platforms using onboard units, while these approaches do not rely on the remote processing of the data.

In this communication paper, a solution for the real-time generation of exploration maps using a low-cost UAV is proposed. This relies on the information provided by a Tello Edu (https://www.ryzerobotics.com/tello-edu accessed on 16 December 2021) drone that streams the images on an external laptop for real-time processing. The solution runs a monocular visual SLAM to register the images and create an unscaled map of the environment. This information is, however, insufficient to estimate the scale of the scene and does not deliver a dense reconstruction of the environment. To address these issues, the SLAM algorithm is then integrated with a CNN-based (Convolutional Neural Network) approach, exploiting a Single Image Depth Estimation (SIDE) algorithm [16] to estimate the scale of the environment and densify the 3D reconstruction of the environment from a set of video frames. This information is finally fused to generate an exploration map and to define the volumes of the environment in real-time. The goal of this approach is to deliver quick 3D maps to support FRs in the exploration of unknown indoor environments during Search and Rescue operations.

The paper is organized as follows. The relevant literature is presented in Section 2, while the proposed methodology is described in Section 3. Tests and results are reported in Section 4, while the pros and cons of the presented approach are discussed in Section 5. Finally, the conclusions and future developments of this work are drawn in Section 6.

2. Background

In this section, a brief overview of the following research topics debated in this paper is given: the monocular SLAM (Section 2.1), the scale estimation of image sequences (Section 2.2), the depth estimation from single images (Section 2.3), and different typologies of exploration maps (Section 2.4).

2.1. Monocular SLAM

SLAM has been an active research topic for many years and, more recently, has gained attention thanks to its use for tasks such as autonomous driving in the automotive industry [19], localization in augmented reality [20], and many other indoor localization applications. Different typologies of SLAM approaches can be distinguished according to the typologies of the sensors used to retrieve the position and map the spaces: solutions integrating visual (RGB or thermal), inertial, and distance (LiDAR or ultrasound) sensors are available in the literature [21]. However, great interest is still given to visual SLAM (vSLAM) approaches where monocular or stereo images are used in real-time to simultaneously construct a map of the environment and localize the cameras. In this paper, we will only focus on monocular visual SLAM, as it currently offers the lightest-weight solution to be embedded on a small UAV. The two major state-of-the-art methods for visual monocular SLAM are feature-based and direct-based algorithms. Feature-based
methods function by extracting a set of unique features from each image. These features are unique points (also called key-points) that can be identified in an image. By matching the same points in multiple views, these algorithms can determine the different positions from which the points were observed [22]. Direct methods do not rely on local regions of the image, but compare the entire images to referenc them, and use image intensities to obtain information about the location and surroundings [23]. An advantage of direct methods is that they generate a denser representation of the environment than feature-based methods. One of the most robust and complete feature-based SLAM implementations is ORB-SLAM and its upgrades [22], which uses the tracked features in core tasks, such as motion estimation, 3D map generation, re-localization when tracking is lost, and the detection of re-observed areas (so-called loop closures), making the system more efficient, reliable, and straightforward. A downside, however, is that it produces very sparse point clouds since, as mentioned previously [24], the goal is long-term and globally consistent localization instead of building a detailed dense reconstruction. Some studies have already tried to densify the point cloud delivered by ORB-SLAM [25], but their results are still unsuitable for navigation and path planning.

LSD-SLAM and its evolutions [26,27] are one of the best-performing, open-source implementations of direct monocular SLAM. It shows very reliable results, and being a direct method produces denser point clouds. This method tracks the motion of the camera towards reference keyframes and, at the same time, estimates semi-dense depth at high gradient pixels in the keyframes. An advantage of the direct method is that the semi-dense depth maps are more suitable for navigation purposes. However, they are still unable to reconstruct low-gradient areas, such as flat walls, leaving room for improvement. According to the literature [28], the accuracies of the two methods are comparable, although feature-based methods appear to be less dependent on the quality of the camera and the illumination changes and, therefore, are more suited for indoor environments.

2.2. Scale Estimation

Visual SLAM approaches are not able to scale the scene using just image information, so several alternative methods have been proposed in the literature. The most common solution is Visual Inertial SLAM (VI-SLAM), where IMU measurements serve as odometry and are fused with the SLAM measurements, often utilizing a Kalman filter [29]. This has the advantage of being less reliant on vision: when vision cannot track features, due to an unlit area, or a sudden camera movement, this can be compensated for by the odometry information. Most of these approaches [24] can use inertial information to scale the scene using the data collected by this instrument.

Alternatively, a simple, and frequently implemented solution, is using ground truth markers or ground control points [30], as in a conventional photogrammetric approach. These markers have easily recognizable patterns (for example, a checkerboard) of known size and, sometimes, known location. Image vision is then used to detect these markers in the scene and scale the world accordingly. A downside of this approach, however, is that it requires the inspection of the scene beforehand to place the markers in the area, limiting its usefulness in many applications. Another method is to utilize single depth measurements delivered by laser or sonar sensors [31]. Since the scale of the SLAM maps is the same in all dimensions, the depth measurement of these instruments can be used to correct the scale of the scene [31].

All these methods require additional sensors, while new research has been conducted in leveraging depth-estimating neural networks to improve the performance of monocular SLAM systems. A previous study [31] exploits a CNN to estimate the depth of the scene and determine the scale in each keyframe. Following the same rationale, the depth information is integrated into a classical SLAM approach in [32], using LSD-SLAM for the tracking of features.
2.3. Single Image Depth Estimation (SIDE) Algorithms

Depth estimation from single images has been an active research topic in the last two decades [33], but has been greatly boosted by the recent development of deep-learning techniques that have overcome most of the limitations of traditional methods. SIDE methods have great importance in many application fields such as autonomous navigation and driving, as well as target tracking or collision avoidance [34].

In the deep-learning domain, different typologies of architectures can be used such as CNN, RNN (Recurrent Neural Network), or GAN (Generative Adversarial networks) according to the requested input, the available data, and the considered application. Most of the available methods have been conceived to use terrestrial indoor and outdoor data, although a growing number of contributions are adopting airborne (mostly UAV) images. The developed algorithms can be further divided into: (i) supervised methods [35], when depth is estimated from images using existing 3D information (DSM or similar) as ground truth; (ii) unsupervised methods, when 3D information is generated with stereo-pairs to avoid the use of extensive datasets [36]; and (iii) semi-supervised methods, when other sources of information (such as LiDAR or synthetic data) are used as proxies to support the depth estimation [37]. SIDE algorithms can be also combined with other tasks, such as semantic segmentation and camera pose [16] to improve the quality of the 3D reconstruction with the support of the other tasks. For a more complete overview of SIDE algorithms, please refer to [34].

2.4. Exploration Maps

The point clouds generated by SLAM and SIDE algorithms can be used to generate obstacle maps. Two typologies of maps can be used for this purpose: mesh or voxel maps. Although mesh maps are largely adopted for 3D model representations, voxel maps are largely used for path planning, as they are computationally efficient and adaptable to low-resolution data [38]. In particular, among the voxel maps, OctoMap [39] is the most widely adopted solution to store the maps and plan the paths.

3. Methodology

The proposed methodology aims at the generation of an exploration map in real-time using the images streamed from a UAV during the flight. This solution can be divided into four main steps, as shown in Figure 1: (i) UAV data acquisition and streaming; (ii) SLAM algorithm; (iii) densification and scaling; and (iv) exploration map generation. These steps are embedded in a ROS (Robotic Operative System) to allow for efficient communication among the different components: the algorithm in (ii) runs on the CPU, while both algorithms in (iii) and (iv) exploit the GPU capability of the used PC. As the SLAM system is feature-based, the map consists of sparse, triangulated 3D feature points, but does not provide us with the correct scale. A CNN processes the RGB images (only a few keyframes) acquired during the flight to retrieve the scale and create dense depth maps. The information produced by the SLAM and the CNN algorithms is then combined to generate a 3D occupancy map.
3.1. Drone and Data Streaming

The Tello EDU has been adopted for data acquisition and streaming (Table 1). This platform is relatively easy to use, has a limited size, and its cost guarantees high replicability, thus addressing First Responders’ needs. A wifi communication allows streaming the images on a laptop, while the available SDK guarantees the reception of the telemetry and the connection with the platform. Although the drone has an onboard IMU, its SDK does not allow for the reading of raw sensor data.

Table 1. Technical specification of the Tello Edu drone.

| Tello Edu | Photo: 5MP (2592 × 1936) |
|-----------|--------------------------|
| Camera    | Video: 720 p, 30 fps      |
| FOV       | 82.6°                    |
| Flight time | 13 min                  |
| Remote control | 2.4 GHz 802.11 n WiFi |
| Weight    | 87 g                     |

3.2. SLAM Algorithm

The SLAM system is based on ORB SLAM 2 [24], which combines well-known ORB [40] descriptors and FAST detectors [41] in an image pyramid structure to allow for a more reliable tracking of the features. This algorithm has shown to be a reliable and well-documented solution in many applications. It consists of three main threads running in parallel and handling separate tasks. The tracking thread takes new images in and uses them to estimate the new position, while the local mapping and the loop closure threads are responsible for building and optimizing the generated map, respectively. Compared to the original implementation [23], we decided to use only six pyramid levels for the feature detection; two keyframes were considered connected if at least 25 features were correctly shared in the object space. Given the irregular movements of the UAV and the relatively low framerate used, we reduced the minimum interval between keyframes to 15 frames.

For each keyframe, the sparse features generated by SLAM are published on the ROS interface. The triangulated positions of these points, and their corresponding x and y pixel coordinates, are used to construct an unscaled sparse depth image that is used as an input by the CNN (as described in Section 3.3).
3.3. CNN-Based Densification and Scaling

A Single Image Depth Estimation (SIDE) network has been developed to estimate the distance of each image pixel in the object space. It has been developed to run in real-time on a consumer laptop after being trained using depth images with real distances. This has a twofold motivation: (i) to scale the reconstruction delivered by SLAM in the object space; and (ii) to densify the 3D reconstruction by fusing the sparse depth map generated by SLAM with the CNN. Different from many SIDE algorithms, the sparse depth samples defined by SLAM are added to the RGB images in input as an additional constraint to refine the depth estimation. The developed method is based on [42], which combines these inputs to predict a detailed depth image (Figure 2). This architecture has an encoder–decoder structure, where the encoder processes the input information and converts it into feature maps. These maps are then stacked together to produce the output of the decoder.

![Figure 2. The architecture of the CNN used [42]. Green boxes describe the encoding layers, while yellow boxes are the decoding layers. Please note that both images and depth maps are initially down-sampled.](image)

The encoding layer of the network is based on the well-known ResNet-50 [43]. The last average pooling layer and linear transformation layer of the encoding at the end of ResNet are replaced with a new depth decoding layer designed by [35]. This depth decoding layer uses an up-sampling strategy with up-projection blocks. Chaining these up-projection blocks allows high-level information to be more efficiently passed forward in the network, while progressively increasing feature map sizes. This enables the construction of their coherent, fully convolutional network for depth prediction with a relatively low number of weights.

Previous experiments [35,42] have shown that $L_1$ (see Equation (1)) as a loss function produced the best results on the RGB-based depth prediction problems, minimizing the error of all the absolute differences between the estimated depth and the ground truth (where $y$ and $\hat{y}$ refer to the real and predicted depth).

$$L_1 = \text{mean}(|y - \hat{y}|)$$

Once trained, the network is initialized using a sequence of different keyframes at the beginning of each acquisition. During this process, the network gets the RGB and an empty set of depth samples. The estimated depth map is then compared with the triangulated SLAM features to estimate the scale factor (see Section 3.3.2) that is then used for the rest of the acquisition to scale the scene to the real size. The CNN is then used to deliver a depth map for each considered keyframe.

3.3.1. CNN Training

The network has been trained using the NYU Depth v2 [44] dataset. This dataset contains 48,521 indoor images, and it has been further increased with data augmentation: rotations, flipping, and random noise on the depth maps have been applied to further prevent overfitting. This dataset depicts indoor scenes and has been recorded with a Kinect camera at a resolution of 640 $\times$ 480 pixels for images and labelled depth images. It includes different types of rooms such as basements, bathrooms, bedrooms, offices, and dining rooms. The random error of Kinect depth measurements increases quadratically with
range, reaching 4 cm at 5 m [45], which can be considered the maximum acceptable range distance delivered by this sensor.

The network learns and infers depths by fusing global and contextual information extracted from the corresponding receptive fields on the images. By doing so, it implicitly embeds the intrinsic parameters of the camera, such as pixel size, focal length, and field of view. However, Kinect’s and Tello’s cameras have different geometries that need to be considered and opportunely compensated for before training the network [46]. The depth map given by the NYU Dept v2 dataset was therefore resampled (and then cropped to consider the different field of view) using the intrinsic parameters reported in Equation (2), where: \( f \) is the original focal length in the x and y are the directions of the frame, \( c \) is the principal point, and \( r \) is the ratio that allows for the correction of the different geometries of the two cameras. Please note that the principal point was set in the centre of the image, while other distortions were not considered in this process.

\[
A_{\text{nn}} = \begin{bmatrix}
  f_x r_x & 0 & c_x r_x \\
  0 & f_y r_y & c_y r_y \\
  0 & 0 & 1
\end{bmatrix}
\]  

During the training, the network uses down-sampled RGB images and the corresponding sparse depths as the input and assesses them using the complete depth images as ground truth. While the 3D reconstruction considers all the points of each image, the scale estimation is performed considering only a sparse subset of points and their corresponding depths. These points are extracted on the RGB image by a FAST detector that emulates the features selected by the adopted SLAM algorithm. It must be noted that FAST is a corner detector and, as such, detects points mainly in correspondence of radiometric discontinuities in the images, describing the salient parts of the scene. A downside is that areas without texture do not provide many feature matches, making it harder to get depth estimates in these areas, and resulting in a lower quality of the reconstruction. The detector can then extract thousands of points per image. Only a limited number of points have been considered in the training process: training with 100, 200, and 400 points per image in input has been used to assess the network performances (see Section 4.1).

### 3.3.2. Scale Initialization

The 3D reconstruction generated by SLAM is scaled using the information of the trained CNN. The arbitrary scale of the visual SLAM algorithm is corrected using a scale ratio. This scale, \( s \) (see Equation (3)) is computed considering the vectors of the depths delivered by FAST features, \( (D_{\text{orb}}) \), and the corresponding pixel depth values determined by the CNN, \( (D_{\text{nn}}) \). During this initialization procedure, the sparse depth map is not used as input for the CNN. Only regions within a 5-m distance are used in this process to consider the limited quality of the Kinect dataset used for training. Next, a median filter is adopted to robustly determine the scale ratio from the noisy estimates generated during the initialization. Only the \( l \) inlier values are used determining the final \( s \) factor, by minimizing square errors according to Equation (4).

\[
s = \frac{D_{\text{orb}}}{D_{\text{nn}}}
\]

\[
s_{\text{orb}} = \argmin \sum_{i=0}^{l} \delta_i \left( sD_{i,\text{orb}} - D_{i,\text{nn}} \right)^2
\]

From experimental tests, it was noticed that 50 keyframes were always able to deliver stable and reliable scale factors. This initialization procedure lasts a few seconds at the beginning of the flight and is compatible with the needs of First Responders.
3.4. Exploration Map

The depth estimations of neural networks often deliver inaccurate reconstructions in correspondence of depth discontinuities (so-called “mixed pixels”) and on the border of the images, especially in correspondence of low textured regions (Figure 3). These artefacts degrade the quality of the point cloud, especially when different depth maps are fused to generate a full 3D reconstruction.

![Figure 3. (a) Example of depth filtering; blue points are preserved while red points are considered mixed pixels and discarded. (b) Example of convex hull selection area: only the region inside the red line is used for depth estimation, the remaining peripheral areas are discarded.](image)

Two different approaches are sequentially applied to limit the areas affected by wrong reconstructions. Mixed pixels are filtered with computationally efficient outlier removal. All the points are initially stored in a KD-tree and the mean distance from each point to its neighbours (i.e., 30 points) is determined. The average and the standard deviation of these distances are then computed for the whole point cloud. The points that have a distance higher than a certain threshold are then removed (red points in Figure 3a). Low-textured areas on the border of the images are removed by just considering the convex hull defined by the SLAM features in the image and excluding the 3D reconstruction outside of this region (see Figure 3b).

The depth maps generated in each keyframe are integrated into a unique exploration map. For this purpose, the point clouds generated using different images are initially converted into point clouds and then registered together (knowing their relative poses from SLAM) in the scaled reference system. The point clouds are then converted into an Octomap that keeps track of the probability of each leaf of the node to be occupied through probabilistic occupancy grid mapping [47]. Using Equation (5), the probability that voxel \( n \) is occupied, given the measurement \( z_t \), is denoted by the log probability term \( P(n|z_{1:t}) \), and the result of the previous estimate is denoted as \( P(n|z_{1:t-1}) \). This allows for the consideration of the probabilities and noise of the input point cloud with a uniform prior probability \( P(n) = 0.5 \). A node is therefore considered occupied if the probability is \( P(n) > 0.5 \) and free otherwise.

\[
P(n|z_{1:t}) = \left[ 1 - \frac{1 - P(n|z_t)}{P(n|z_t)} \right] \frac{1 - P(n|z_{1:t-1})}{P(n|z_{1:t-1})} \frac{P(n)}{1 - P(n)}
\]  

(5)

In the implementation, Equation (5) has been simplified using log-odds notation to make it faster (see Equations (6) and (7)).

\[
L(n|z_{1:t}) = L(n|z_{1:t-1}) + L(n|z_t)
\]  

(6)

\[
L(n) = \log \left[ \frac{P(n)}{1 - P(n)} \right]
\]  

(7)

Two couples of parameters need to be defined to set the Octomap up: (i) the \( l_{\text{min}}/l_{\text{max}} \) threshold and (ii) \( \text{Hit}/\text{Miss} \). The first couple defines the upper and lower bounds of the occupancy: it gives the number of updates needed to change the state of a voxel (in
Equation (6)). As an example, decreasing \( l_{max} \) and increasing \( l_{min} \) generate faster updates of the exploration map, at the cost of higher noise in the map. In this case, a voxel, observed for a long time, won’t need to be observed as free for a similar amount of time to be considered a free area. The min/max values are used to set the limits in Equation (8).

\[
L(n|z_{1:t}) = \max(\min(L(n|z_{1:t−1}) + L(n|z_t), l_{max}), l_{min})
\]  

(8)

On the other hand, \( Hit \) values give the probability \( P(n|z_t) \) when a voxel is occupied, while \( Miss \) give the same probability when the voxel is registered as free. High parameter values mean higher trust on the quality of the 3D reconstruction. From experimental evidence, \( Hit = 0.7, Miss = 0.4, l_{min} = 0.12, \) and \( l_{max} = 0.97 \) showed better results and were adopted in the following tests.

4. Tests and Results

This section summarizes the most meaningful tests performed. In Section 4.1, the preliminary tests to determine the optimal set up of parameters are presented. In Section 4.2, two examples of 3D reconstructions generated by UAV flights are shown to assess the quality of results achievable by the presented methodology.

4.1. Network Training

The network has been trained using the dataset described in Section 3.3.1. From experimental evidence, the number of epochs was set to 15; the learning rate started by 0.01 and was reduced by 20% every 5 epochs, while a weight decay of \( 10^{-4} \) was applied for regularization as suggested in [43]. Several tests have been performed considering different sampling densities (i.e., number of FAST features extracted), as discussed in the following section.

4.1.1. Sampling Density

Different tests have been performed to assess the optimal number of sparse features to use as input in the CNN. A high number of features provides more details of the 3D environment and could determine a better 3D reconstruction. On the other hand, a higher computation time is needed to process this larger number of points. The optimal value to use was determined by comparing the residuals of the computed depth map with the ground truth on the whole scene (Table 2). For this purpose, 350 sample images of NYU Depth v2 [44] were processed with a consumer-grade PC (Lenovo, ThinkPad 8GB RAM, Intel Core i5-560M).

| Points | MAE [m] | Abs rel | RMSE [m] | \( \delta_{10} \) | Time [s] |
|--------|---------|---------|----------|--------------|---------|
| 0      | 0.413   | 0.157   | 0.562    | 0.463        | 0.02    |
| 100    | 0.165   | 0.058   | 0.279    | 0.852        | 0.02    |
| 200    | 0.146   | 0.051   | 0.252    | 0.878        | 0.046   |
| 300    | 0.137   | 0.047   | 0.232    | 0.899        | 0.045   |
| 400    | 0.119   | 0.042   | 0.212    | 0.907        | 0.039   |

Table 2 shows the results achieved when varying the number of features from 0 to 400. The 0-points case has large residuals, while results in other tests are still compatible with the expected quality of the point cloud in emergency conditions. Raising the number of features improves the quality of the generated point cloud, although this is marginal
compared to the increase in the processing time. Overall, 100 points looked to be a good compromise among results quality, processing time, and minimum expected number of features extracted by the SLAM algorithm. Therefore, this case was adopted in the following tests.

4.1.2. Scale Estimation

The initialization procedure of the network estimates the arbitrary scale factor of monocular SLAM. In these tests, the accuracy of this scale estimation is evaluated by comparing it to the ground truth. In order to assess the results independently from the training dataset used, the tests were performed on the TUMRGB-D benchmark dataset [48]. Thanks to this dataset, it was possible to assess the capability of this method to generalize good scale estimations in different environments.

The results (Table 3) show that the scale factor generated by the CNN without filtering is not accurate, while the implemented median filter efficiently removes the wrong estimates, thus delivering factor scales very close to the ground truth.

Table 3. Scale factors were computed on three different subsets (acquired in three different locations) of the TUMRGB-D dataset. The ground truth scale factors, the unfiltered average values computed by the CNN, and the filtered ones are provided.

| Dataset | Ground Truth | CNN   | Median Filter |
|---------|--------------|-------|---------------|
| TUM1    | 2.43         | 2.31  | 2.47          |
| TUM2    | 2.16         | 1.53  | 2.09          |
| TUM3    | 1.29         | 1.35  | 1.31          |

4.2. 3D Reconstruction of Test Environments—First Results

The developed algorithm was tested in two different environments of relatively limited extension (max 8 m length), characterized by different shapes and variable surface textures. Although these environments (i.e., offices complete of furniture) differ from a classical rescue scenario faced by FR, they can give meaningful information about the capability of the presented approach to reconstruct and scale an unknown scene.

In these tests, the same medium-low performance PC (i.e., Lenovo ThinkPad) has been used to assess the real-time performance of the solution in an operational context. The videos were acquired at 10 fps, as this rate was shown to reliably track features without being too computationally intensive for the PC that was used. The following shows the generated point clouds instead of the exploration maps in order to give better insights into the quality of the 3D reconstruction. An example of voxel maps is shown in the video provided with this paper (see the Supplementary Materials).

4.2.1. First Test

The acquisition lasted about 70 s and it was performed by moving the drone in the room. The rotations of the platform were quite rough (see Supplementary Materials) because of the confined spaces. This made the tracking of the features more challenging, although the SLAM algorithm was still able to run without losing the tracking along the image sequence.

The 3D reconstruction generated in real-time is shown in Figure 4. Several residual deformations are still visible in the point cloud, affecting its quality. The level of noise increases in correspondence of the glasses (upper part of the scene in b) and in the areas with low texture (lower part in c), as was expected. The entrance of the room is also a critical part of the reconstruction (lower left part in c). However, the estimated scale is accurate, delivering realistic measures, while the overall geometry of the scene is preserved, as needed by FR.
Figure 4. (a) Panoramic image of the test environment; (b) perspective view of the generated point cloud; and (c) its comparison to a reference planimetry of the floor.

4.2.2. Second Test

The second test was run in a square room. The acquisition lasted about 50 s. As in the previous case, the noise of the SIDE reconstruction cannot be completely removed, generating some artefacts in the scene (Figure 5). However, both the scale and the shape of the room are preserved. Glasses and reflective surfaces strongly affect the reconstruction, as shown in the right part of the room, and generate significant errors in the reconstruction.

Figure 5. (a) Perspective view of the generated point cloud of the test environment; and (b) its comparison to a reference planimetry of the floor.
5. Discussion

The generated point clouds can delineate the main elements of the indoor environment, while smaller details (such as furniture) are not correctly described because of the noise in the 3D reconstruction. This problem is more evident in the case of glasses and areas with no texture where artefacts (such as waived surfaces) can be easily found. Several elements contribute to this problem: (i) current SIDE algorithms are not able to reconstruct 3D environments as stereo-pair approaches can do, as confirmed by several studies in the literature. More advanced architectures could only partially improve the results, although they are too computationally intensive for the hardware used in this paper; (ii) the network has then been trained, adapting depth images of the Kinect and without accounting for residual camera model deformations. This has led to a lower 3D reconstruction quality, especially on the border of each generated depth map; and (iii) only 100 feature points were chosen to run the tests in real-time. A higher number of features could partially reduce the noise level, at the cost of higher computational costs, as already discussed.

On the other hand, the scale estimation has shown that stable values can be reached after a few seconds of acquisition, being completely compatible with the requirements given by search and rescue scenarios. Longer estimation times would not give tangible improvements, costing more time for the initialization procedure. The scale estimation showed good results in all the performed tests, demonstrating its transferability to different datasets independently from its training.

6. Conclusions and Future Developments

This paper presents the first results of a 3D, real-time mapping solution using a low-cost UAV and combining SLAM and deep learning algorithms. The UAV can explore unknown environments and delivers scaled exploration maps to answer the needs given by First Responders involved in Search and Rescue activities: their main goal is to improve scene awareness before entering indoor spaces.

The first tests showed promising results: the presented approach can reliably define rough exploration maps of the surveyed environment. Although the quality of the generated point clouds is still low and not comparable to photogrammetric reconstructions, the scene is correctly scaled, and the volumes of empty spaces are appropriately reconstructed, showing the potential of this method in many practical scenarios. The envisioned initialization process can run in a few seconds, allowing for the correct scaling of the scene, and the experiments were run in real-time using just a low-performance PC. The SLAM algorithm can reliably track images on the drone, despite the confined spaces and the rough rotations performed by the drone, as it normally happens in an operational condition.

The presented results showed that there is still much room for improvement: future tests will be performed using a higher quality PC to run a deeper network and assess if this allows for better 3D reconstructions. The training dataset was not ideal because of the low quality of the depth maps delivered by Kinect and the need to adapt the data to the UAV’s camera parameters. In this regard, a new dedicated dataset would allow for better training of the network and improved reconstructions.

Considering Search and Rescue scenarios, indoor environments are usually dark with limited/absent illumination. In this regard, the presented work considered “normal” illumination conditions. More work will be needed to tackle the problem considering more critical light conditions. Other elements, such as a semantic understanding of the scene and autonomous navigation of the drone, have not been addressed in this contribution, but could represent useful research extensions to tackle in the future.

Supplementary Materials: The following video of the acquisition and processing is available online at: https://vimeo.com/670912813, while the code is available at: https://github.com/annesteenbeek/sparse-to-dense-ros.
Author Contributions: Conceptualization, A.S. and F.N.; methodology, A.S. and F.N.; algorithm development, A.S.; tests and validation, A.S.; writing, review, and editing, A.S. and F.N.; visualization, A.S.; supervision, F.N. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References
1. Nex, F.; Duarte, D.; Steenbeek, A.; Kerle, N. Towards Real-Time Building Damage Mapping with Low-Cost UAV Solutions. Remote Sens. 2019, 11, 287. [CrossRef]
2. Li, F.; Zlatanova, S.; Koopman, M.; Bai, X.; Diakité, A. Universal Path Planning for an Indoor Drone. Autom. Constr. 2018, 95, 275–283. [CrossRef]
3. Sandino, J.; Vanegas, F.; Maire, F.; Caccetta, P.; Sanderson, C.; Gonzalez, F. UAV Framework for Autonomous Onboard Navigation and People/Object Detection in Cluttered Indoor Environments. Remote Sens. 2020, 12, 3386. [CrossRef]
4. Khosianwan, Y.; Park, Y.; Moon, I.; Nilakantan, J.M.; Nielsen, I. Task Scheduling System for UAV Operations in Indoor Environment. Neurocomput. Appl. 2018, 31, 5431–5459. [CrossRef]
5. Nex, F.; Armenakis, C.; Cramer, M.; Cucci, D.A.; Gerke, M.; Honkavaara, E.; Kukko, A.; Persello, C.; Skaloud, J. UAV in the Advent of the Twenties: Where We Stand and What Is Next. ISPRS J. Photogramm. Remote Sens. 2022, 184, 215–242. [CrossRef]
6. Zhang, N.; Nex, F.; Kerle, N.; Vosselman, G. LISU: Low-Light Indoor Scene Understanding with Joint Learning of Reflectance Restoration. ISPRS J. Photogramm. Remote Sens. 2022, 183, 470–481. [CrossRef]
7. Xin, C.; Wu, G.; Zhang, C.; Chen, K.; Wang, J.; Wang, X. Research on Indoor Navigation System of UAV Based on LIDAR. In Proceedings of the 2020 12th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA), Phuket, Thailand, 28–29 February 2020; pp. 763–766.
8. Lin, Y.; Hyyppä, J.; Jaakkola, A. Mini-UAV-Borne LIDAR for Fine-Scale Mapping. IEEE Geosci. Remote Sens. Lett. 2011, 8, 426–430. [CrossRef]
9. Pu, S.; Xie, L.; Ji, M.; Zhao, Y.; Liu, W.; Wang, L.; Zhao, Y.; Yang, F.; Qiu, D. Real-Time Powerline Corridor Inspection by Edge Computing of UAV LIDAR Data. ISPRS Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci. 2019, 42, 547–551. [CrossRef]
10. De Croon, G.; De Wagter, C. Challenges of Autonomous Flight in Indoor Environments. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 1003–1009.
11. Falanga, D.; Kleber, K.; Mintchev, S.; Floriano, D.; Scaramuzza, D. The Foldable Drone: A Morphing Quadrotor That Can Squeeze and Fly. IEEE Robot. Autom. Lett. 2019, 4, 209–216. [CrossRef]
12. Amer, K.; Samy, M.; Shaker, M.; Elhelw, M. Deep Convolutional Neural Network Based Autonomous Drone Navigation. In Proceedings of the Thirteenth International Conference on Machine Vision, Rome, Italy, 2–6 November 2020; Osten, W., Zhou, J., Nikolaev, D.P., Eds.; SPIE: Rome, Italy, 2021; p. 46.
13. Arnold, R.D.; Yamaguchi, H.; Tanaka, T. Search and Rescue with Autonomous Flying Robots through Behavior-Based Cooperative Intelligence. J. Int. Humanit. Action 2018, 3, 18. [CrossRef]
14. Bai, S.; Chen, F.; Englot, B. Toward Autonomous Mapping and Exploration for Mobile Robots through Deep Supervised Learning. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 2379–2384.
15. Chakravarty, P.; Kelchtermans, K.; Roussel, T.; Wellens, S.; Tuylstraets, T.; Van Eycken, L. CNN-Based Single Image Obstacle Avoidance on a Quadrotor. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 6369–6374.
16. Madhuanand, L.; Nex, F.; Yang, M.Y. Self-Supervised Monocular Depth Estimation from Oblique UAV Videos. ISPRS J. Photogramm. Remote Sens. 2021, 176, 1–14. [CrossRef]
17. Knobelreiter, P.; Reinbacher, C.; Shekhovtsov, A.; Fock, T. End-To-End Training of Hybrid CNN-CRF Models for Stereo. In Proceedings of the Conference of the IEEE Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017.
18. Yang, M.Y.; Kumaar, S.; Lyu, Y.; Nex, F. Real-Time Semantic Segmentation with Context Aggregation Network. ISPRS J. Photogramm. Remote Sens. 2021, 178, 124–134. [CrossRef]
19. Singandhupe, A.; La, H.M. A Review of SLAM Techniques and Security in Autonomous Driving. In Proceedings of the 2019 Third IEEE International Conference on Robotic Computing (IRC), Naples, Italy, 25–27 February 2019; pp. 602–607.
20. Saeedi, S.; Spink, T.; Gorgovan, C.; Webb, A.; Clarkson, J.; Tomusk, E.; Debrunner, T.; Kaszyk, K.; Gonzalez-De-Aledo, P.; Rodchenko, A.; et al. Navigating the Landscape for Real-Time Localization and Mapping for Robotics and Virtual and Augmented Reality. Proc. IEEE 2018, 106, 2020–2039. [CrossRef]
21. Stachniss, C.; Leonard, J.J.; Thrun, S. Simultaneous Localization and Mapping. In Springer Handbook of Robotics; Springer: Berlin/Heidelberg, Germany, 2016; pp. 1153–1176. ISBN 978-3-319-32552-1.
22. Campos, C.; Elvira, R.; Rodriguez, J.J.G.; Montiel, J.M.M.; Tardós, J.D. ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial and Multi-Map SLAM. IEEE Trans. Robot. 2020, 37, 1874–1890. [CrossRef]
47. Wurm, K.M.; Hornung, A.; Bennewitz, M.; Stachniss, C.; Burgard, W. Octomap: A Probabilistic, Flexible, and Compact 3D Map Representation for Robotic Systems. In Proceedings of the Autonomous Robots. 2010. Available online: https://www.researchgate.net/publication/235008236_OctoMap_A_Probabilistic_Flexible_and_Compact_3D_Map_Representation_for_Robotic_Systems (accessed on 30 January 2022).

48. Sturm, J.; Engelhard, N.; Endres, F.; Burgard, W.; Cremers, D. A Benchmark for the Evaluation of RGB-D SLAM Systems. In Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, Vilamoura-Algrave, Portugal, 7–12 October 2012; pp. 573–580.