2024 International Conference on Unmanned Aircraft Systems (ICUAS)
June 4-7, 2024. Chania, Crete, Greece

Visual Environment Assessment for Safe Autonomous Quadrotor Landing

Mattia Secchiero[1,2], Nishanth Bobbili[1], Yang Zhou[1], and Giuseppe Loianno[1]

***Abstract*— Autonomous identification and evaluation of safe landing zones are of paramount importance for ensuring the safety and effectiveness of aerial robots in the event of system failures, low battery, or the successful completion of specific tasks. In this paper, we present a novel approach for detection and assessment of potential landing sites for safe quadrotor landing. Our solution efficiently integrates 2D and 3D environmental information, eliminating the need for external aids such as GPS and computationally intensive elevation maps. The proposed pipeline combines semantic data derived from a Neural Network (NN), to extract environmental features, with geometric data obtained from a disparity map, to extract critical geometric attributes such as slope, flatness, and roughness. We define several cost metrics based on these attributes to evaluate safety, stability, and suitability of regions in the environments and identify the most suitable landing area. Our approach runs in real-time on quadrotors equipped with limited computational capabilities. Experimental results conducted in diverse environments demonstrate that the proposed method can effectively assess and identify suitable landing areas, enabling the safe and autonomous landing of a quadrotor.**

Supplementary Material

Video: https://youtu.be/3tH621vF8LM

I. Introduction

Unmanned Aerial Vehicles (UAVs) have become increasingly popular platforms to assist humans in several complex and dangerous applications such as surveillance, law enforcement, mapping, search and rescue, delivery services and precision agriculture [1], [2]. The development of novel autonomous algorithms coupled with the drop in price–performance ratio of processors and sensors supported also the execution of complex tasks such as collaborative transportation [3], autonomous flight [4], collision avoidance [5], exploration [6] as well as shipping and delivery [7] or industrial inspection [8]. To ensure the safety of individuals, structures, and overall mission success in the aforementioned applications it is commonplace to equip aerial robots with intelligent landing capabilities [9]–[11]. These mitigate the risks posed by mechanical and sensory failures, ensuring secure operations in challenging scenarios. This not only minimizes potential threats posed to individuals and structures, but also enables the successful execution of tasks that require such capabilities. For example, in the case of a drone delivery system that operates in urban environments, the ability to accurately identify suitable landing zones becomes crucial for the successful delivery of packages. In agriculture, drones play a pivotal role in monitoring crops, assessing plant health, and optimizing agricultural practices. Upon completing these tasks, the drone requires a reliable and safe landing procedure.

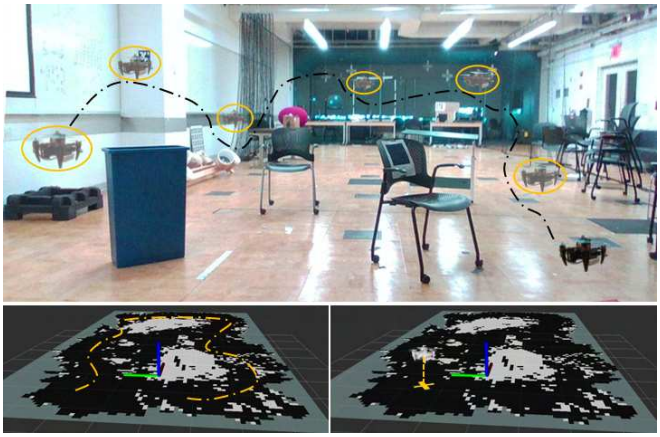


Fig. 1: *Top*: our drone navigating and mapping the environment. *Bottom*: the associated 2D binary map of the safe and unsafe landing locations (*left*) and the chosen safe landing spot in the 2D map (*right*). The black areas are the safe landing locations, while the gray ones are unsafe.

However, current state-of-the-art solutions tend to be fragile and computationally intensive, often require preliminary environment knowledge and still offer limited autonomy. Similarly, the majority of commercially available landing solutions often relies on manually designed, pre-defined navigation policies to aid humans in guiding the robots to land near the intended or required location, therefore offering minimal or no autonomy also for the landing process.

This paper presents several significant contributions. First, we propose a novel visual environment detection and assessment approach for the safe autonomous landing of aerial robots. Compared to the state-of-the-art solutions, our method efficiently combines metric and semantic information, leveraging both RGB images and the disparity maps extracted from the environment. Specifically, the 3D points of the environment, generated from the disparity map, are projected onto the segmented RGB image to efficiently process only the ones associated with safe regions, obtaining an effective and efficient solution. Furthermore, our method does not need to build and store computationally intensive

[1]The authors are with the New York University, Tandon School of Engineering, Brooklyn, NY 11201, USA. email:{ms15143, nb3553, yangzhou, loiannog}@nyu.edu.

[2]The author is with the Department of Information Engineering, University of Padua, 35131 Padua, Italy. email:{mattia.secchiero}@studenti.unipd.it.

This work was supported by the NSF CAREER Award 2145277, the DARPA YFA Grant D22AP00156-00, Qualcomm Research, Nokia, and NYU Wireless.

2024 International Conference on Unmanned Aircraft Systems (ICUAS) | 979-8-3503-5788-2/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICUAS60882.2024.10557078

elevation maps. Instead it directly generates and updates a 2D binary map of safe and unsafe landing zones, without sacrificing any relevant information compared to existing approaches. Second, we define several cost metrics based on critical geometric attributes such as slope, flatness, and roughness, extracted from the visual information to assess potential viable landing areas. Finally, our pipeline operates on-board, without relying on any off-board streaming of data, GPS or pre-obtained information. We demonstrate that our framework successfully enables safe quadrotor landings in multiple and challenging indoor environments.

II. Related Works

Several works address the problem of identifying a safe landing zone and implementing autonomous landing procedures. The vast majority of the existing approaches are vision-based, given the Size, Weight, and Power (SWaP) constraints of small-size aerial robots. For example in [12] the authors compute the local slope and roughness of a Digital Elevation Model (DEM) to detect and avoid hazards such as steep slopes, rocks, cliffs, and gullies. Another approach, as described in [13], defines a cost function that evaluates the physical properties of the local neighborhood within an elevation map region to identify safe landing areas, using a robot-centric fixed size map. This, however, confines the environmental knowledge exclusively to the region beneath the drone. The solution proposed in [10] instead infers a safe landing zone by evaluating slope and roughness from the DEM of the environment, obtained using a Structure from Motion (SfM) algorithm. Conversely, the authors in [9] apply a slope and roughness threshold to the DEM image gradient, to only retain flat terrains. Since the elevation maps are pre-determined, a Neural Network (NN) segments the RGB images of the possible landing area as a final evaluation step. The semantic information is used to asses the validity of the area and to overcome the fact that the maps could be outdated. Other works such as [11] and [14] directly implement semantic segmentation on a DEM. Segmentation is employed to classify hazardous and safe landing locations without resorting to plane fitting techniques or gradient thresholding. However, these approaches rely on pre-collected LiDAR data to get the elevation models, which may not be suitable for small-size UAVs and might not always have access to up-to-date maps.

Yet, it is essential to develop general-purpose solutions that do not rely on GPS [9], [12] or pre-determined maps [9], [11], [14]. The former would be unsuitable for indoor or GPS-denied environments, while the latter's reliance on predefined environment poses a challenge in dynamic settings, potentially leading to catastrophic outcomes. [13] and [10] further illustrate this by employing a fixed-size map, limiting environmental awareness to the area directly below the drone, thus disregarding a big portion of the overflown area. In [15], the 3D information are lacking, consequently failing to comprehensively address crucial factors like slope, flatness, and roughness.

Compared to the aforementioned existing solutions, we directly construct a variable dimension 2D binary map to guide the drone toward a safe landing location. In such a way, our approach does not need to derive any elevation map, resulting in a lighter and more efficient implementation, while still evaluating all the relevant aspects related to the safe site detection. In addition, our pipeline stands out by its independence from external aids such as GPS, off-board or pre-obtained geometric information and autonomously implements inspection and landing behaviours.

III. Methodology

Figure 2 gives an overview of the entire system's pipeline. The drone leverages a stereo camera pair combined with data coming from an Inertial Measurement Unit (IMU) to compute Visual-Inertial Odometry (VIO). The real-time estimation of the drone's position and orientation, with respect to the fixed world reference frame ($\mathcal{F}_W$), is essential for autonomous flight capabilities such as exploration and autonomous landing. The 2D occupancy grid, on the other hand, is generated considering the segmented RGB images and the disparity maps obtained from a stereo pair. This occupancy grid directly embeds both semantic and geometric information, representing the safe and unsafe landing regions of the environment. At the perception level, one key distinction between our approach and other methodologies lies in the way we generate the map of safe and unsafe landing locations. In most other approaches, the workflow involves the initial construction of a DEM or an elevation map. Subsequently, a binary map of the environment is created, and safe landing areas are identified within this map. In contrast, our method simplifies this process by directly creating a 2D variable-dimension occupancy grid, without the need for elevation maps. This grid encodes a binary classification, distinguishing safe and unsafe landing locations, while still retaining the essential 3D information. Furthermore, it also employs both metric and semantic information, unlike many other approaches that rely solely on one type of information, increasing the robustness of the overall solution. The grid is dynamically updated based solely on the safe point cloud data that meet all the safe site criterion.

Finally, based on this representation, in the evaluation step we find the actual landing zone. From the 2D map and the drone's position we locate the best landing zone by minimizing a cost function that considers *(a)* the drone's distance from a potential safe area and *(b)* the distance of the closest unsafe point to a possible safe area. This process is iterated across the entire map to find a safe zone, large enough to accommodate the drone during landing while also ensuring a safety margin. Once the inspection behaviour concludes, or the necessity of landing arises, the safe landing coordinates are retrieved and the landing behaviour is performed autonomously.

A. Safe Site Detection

1) ***Semantic Information****:* The RGB images are segmented through BiSeNetV2 [16], a real-time semantic seg-

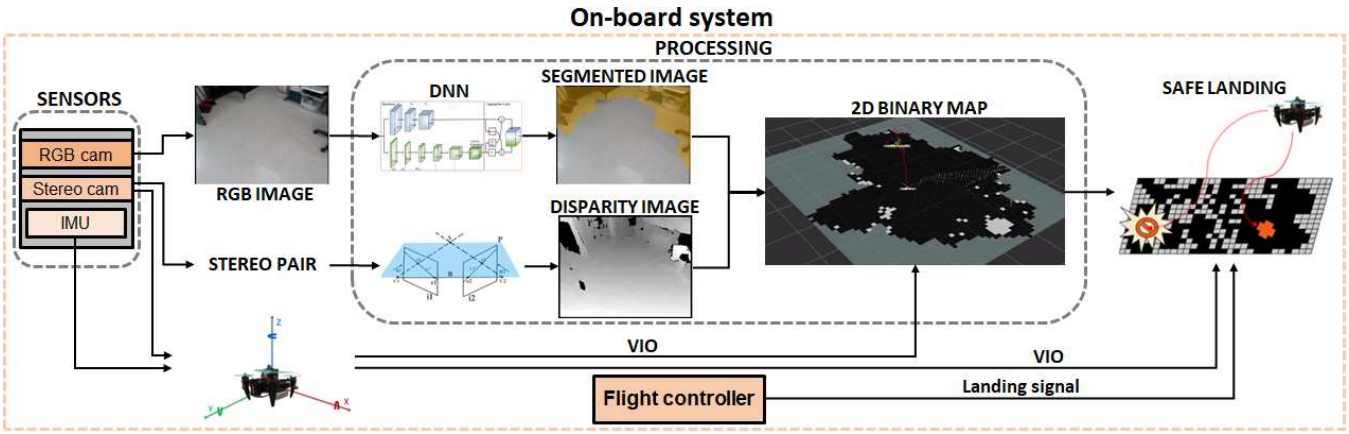


Fig. 2: Overview of our autonomous safe site detection and landing system: we use our quadrotor with a NVIDIA Jetson NX for computation and a stereo camera for VIO & mapping the environment. All our algorithms run in real-time onboard.

mentation neural network. Compared to an encoder-decoder structure or the pyramid pooling modules often used in semantic segmentation, this network proposes a bilateral structure, namely treats the spatial details and categorical semantics separately to achieve high accuracy and high efficiency for real-time segmentation tasks. The features extracted by the two branches of the dual-pathway backbone are then merged together by an aggregation layer. Additionally, to enhance the inference time of the network, we leverage the trained BiSeNetV2 model and we optimize it through the NVIDIA TensorRT library [17]. This significantly accelerates the network's performance and also reduces the model dimensions. The goal of the network is to recognize image regions that are suitable for landing such as grass fields, pavements, roads, floors, etc. For each pixel in the image the NN associates a semantic meaning to it

$$f : (u, v) \mapsto C \tag{1}$$

where at each location (u, v) the pixel is characterized by a label C. Knowing the relation between the label $C \in \mathbb{R}^+$ and the corresponding class (e.g., $C : 0 \mapsto$ safe landing, $C : 1 \mapsto$ people, $C : 2 \mapsto$ obstacles, ...) we are able to infer if the region is suitable for landing.

*2) **Geometric Information**:* From the stereo pair instead we derive a disparity map. Utilizing the camera intrinsic and extrinsic parameters, we calculate the Cartesian coordinates (x, y, and z) of each point, generating the associated point cloud. Subsequently, we re-project the point cloud on the segmented image to only keep the points associated to a safe landing region as

$$\begin{bmatrix} u' \\ v' \\ w \\ * \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R}_{SL}^{RGB} & \mathbf{t}_{SL}^{RGB} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \tag{2}$$

where f_x and f_y represent the focal length and c_x and c_y are the optical center coordinates. Starting from a point (x, y, and z) we determine its corresponding pixel coordinate (u, v), using the transformation equations $u = \frac{u'}{w}$ and $v = \frac{v'}{w}$. It is important to note that in our case, the RGB camera frame ($\mathcal{F}_{RGB}$) and the left stereo camera frame ($\mathcal{F}_{SL}$) are not perfectly aligned. Thus, we account for the rigid transformation between the two frames ($\mathbf{H}_{SL}^{RGB} \in \mathrm{SE}(3)$), as indicated in the second term of the right-hand side of eq. (2). This adjustment ensures the accurate projection of 3D points onto the 2D image. Once we have the pixel coordinates (u, v), we check if the corresponding image location falls within a safe landing region. In the positive case, the 3D point is retained; otherwise, it is flagged as unsafe and discarded. By only considering the safe points we speed-up the computation, improving the performance of the algorithm.

Subsequently, the point cloud is filtered and down-sampled to enhance its quality and suitability for safe landing site detection. The point cloud is initially down-sampled using a voxel grid filter. The "leaf size" parameter controls the voxel size, with larger values resulting in greater down-sampling. In our case, we choose to retain one point each 0.1×0.1 m^2, striking a balance between processing speed and accuracy. Then, a statistical outlier removal filter is applied in order to remove the points that significantly deviate from the heuristic distribution of the point cloud. To further improve the point cloud quality, we employ a Moving Least Square (MLS) smoothing filter, resulting in a smoother point cloud less affected by noise. Finally, a plane fitting algorithm [18] is executed to identify planar regions within the point cloud. This allows to retain only the points associated to a flat surface, that also satisfy predefined slope and roughness thresholds. However, to correctly assess the metric properties of the scene, we first have to ensure that the point cloud is aligned with the world reference frame ($\mathcal{F}_W$), which conveniently coincides with our map reference frame ($\mathcal{F}_{2DM}$). To this end we consider the rigid transformation between the left stereo frame and the world frame $\mathbf{H}_{SL}^{W} \in SE(3)$, since the point cloud is originally aligned with $\mathcal{F}_{SL}$. By leveraging the drone's odometry, we can compute $\mathbf{H}_{SL}^{W}$ and apply the necessary compensation to get the true plane's inclination. Moreover, in this way we also compensate for the drone's Roll, Pitch, and Yaw (RPY) rotation since it can be tilted with respect to the surface. Subsequently, the plane inclination can be computed considering the angle between its normal vector $\hat{\mathbf{n}} = (\mathbf{n_x}, \mathbf{n_y}, \mathbf{n_z})$ and the $\mathbf{z}$-axis of the

world frame (parallel to the gravity vector $\hat{\mathbf{g}}$) as

$$\phi_x = \text{atan2}(\mathbf{z}, \mathbf{n_y}),\ \phi_y = \text{atan2}(\mathbf{z}, \mathbf{n_x}). \tag{3}$$

In our specific setup, any planes with inclinations exceeding $15°$ are considered unsafe. Furthermore, we can establish whether a point qualifies as a plane inlier or not by evaluating its distance to the fitted plane, thus defining the maximum roughness admitted. To retrieve the distance of a point from the fitted plane, we compute the magnitude of the perpendicular vector connecting the point to the plane as

$$\text{dist} = \frac{|Ax + By + Cz + D|}{\sqrt{A^2 + B^2 + C^2}}, \tag{4}$$

where x, y and z are the point's coordinates and A, B, C and D are the plane coefficients. In our case if a point is more than 0.05 m away from the plane it is considered unsafe. The map is a composition of processed images and point clouds that satisfy the safe site criteria detailed in this section.

B. Environment Assessment and Autonomous Landing

1) ***Environment Assessment****:* Up to this point, our knowledge has provided us with a general understanding of the distribution of safe and unsafe points across the map. However, our objective is to pinpoint the optimal landing location. To achieve this, we search for a patch on the map that spans an area approximately 1.85 times the size of the drone. This patch must be a sufficiently large region consisting exclusively of safe points and accounting for an additional safety margin. The patch is considered safe when every cell within it has a safety probability "p" of 95% or higher. This means that we ensure all cells in the designated patch are highly likely to be safe, with safety probabilities ranging from 0% to 100%, where 100% indicates complete safety. The probability "p" of each cell is computed as done in [19]. Once this area is found, we compute the associated cost

$$J = \alpha \cdot J_d + \beta \cdot \frac{1}{J_{un}}, \tag{5}$$

with $\alpha + \beta = 1$ and $\alpha,\ \beta \in [0, 1]$, J_d is the distance between the center of the safe zone and the drone while J_{un} is the distance between the center of the safe zone and the closest unsafe point. J_d and J_{un} are both computed as Euclidean distances.

This approach prioritize landing zones that are not only closer to the drone, but also farther away from unsafe areas. Fine-tuning the parameters α and β enables us to adjust the behavior for identifying the safest landing zone. By increasing α and reducing β, the algorithm tends to find a safe landing zone that is closer to the drone, but potentially nearer to obstacles, and vice versa. Additionally, our final evaluation takes into account the drone's battery consumption, which is influenced by the Euclidean distance between the drone and the 3D landing location, as shown in [20]. By iterating this last step over the whole map we aim to minimize the cost function and only keep the safest landing zone, namely the one associated to the lowest cost possible. Whenever new areas are overflown or the drone changes its position, the environment is re-perceived, the map is updated accordingly and the best safe landing zone is published.

In summary, our safe site detection pipeline employs a comprehensive evaluation of region safety, taking into account both semantic information and geometric information, including flatness, roughness, steepness, a distance transform and the drone size.

2) ***Autonomous Landing****:* The landing step is pretty straightforward and doesn't require any particular process to be involved. Once the landing is required, a minimum snap trajectory generation algorithm [21], [22] is used to find a path from the drone's actual position to the safe landing spot. When initiated, the drone follows this two-step behaviour:

- Fly above the safe landing zone, utilizing the minimum snap trajectory.
- Decrease the height until it reaches the ground level.

This approach is employed since no collision avoidance behaviour has been utilized, thus prioritizing safety and avoiding any possible crash.

IV. Results

To validate the proposed approach, we execute a series of real experiments in a challenging, large indoor environment measuring $26 \times 10 \times 4\ \text{m}^3$, situated at the Agile Robotics and Perception Lab (ARPL, New York University). In particular, our validation process involved two key aspects: *(a)* **environmental changes**, to simulate different evaluation and landing scenarios; *(b)* **waypoints diversification**, to simulate different inspection strategies.

A. System Setup

Our drone is a compact and versatile system, with a diameter of 0.27 m and a weight of 1.1 kg. It is equipped with a PX4 Autopilot flight controller, for high level position control, and a Nvidia Jetson NX computing board. For convenience, without loss of generality of our approach, we employ two stereo cameras to decouple the localization and safe landing evaluation due to the camera characteristics, introducing as well some redundancy in the system. The first one, a RealSense T265 tracking camera, employed to obtain a robust VIO and the second one, responsible for mapping the environment and detecting the safe landing zone. Depending on mission requirements and constraints, it is also possible to effectively operate with a single stereo camera, either pointing directly downward or tilted at a 45-degrees angle, not affecting the approach and results of this work. The system relies on the ROS middleware [23], facilitating communication and integration among the various modules. Considering the real-time and resource-constrained applications, these modules are managed by a nodelet, that reduces computation and latency by sharing memory space and avoiding inter-process communication overhead.

B. Neural Network Training and Evaluation

To train our network, we leverage the ADE20K semantic scene parsing dataset [24], since it provides a wide set of

indoor and outdoor environments, covering a wide range of classes and examples relevant for addressing the safe landing task. However, rather than using the original 150 classes, we have manually clustered them into 11: *water, people/animals, sky, trees, man-made obstacles, nature obstacles, safe landing site, light, vehicles, background, buildings*. In such a way we enhance the inference time simplifying the problem and allowing the segmentor to generalize better, as done in [9]. Our training pipeline is based on the PyTorch framework developed by openMMLab [25]. It consists in an iteration-based training process using a Stochastic Gradient Descent (SGD) optimizer for 160 K iterations, and two NVIDIA GPUs with batch size equal to 8. The optimizer parameters are reported in Table I.

Optimizer parameters	**Value**
Learning Rate	0.05
Momentum	0.9
Weight Decay	0.0005
Decay Type	Polynomial Decay
Polynomial Decay parameters	**Value**
Learning $\text{Rate}_{\min}$	1×10^{-4}
Power	0.9

TABLE I: NN Training parameters

To improve the quality of our segmentation results, we performed a fine-tunig on the NN using a custom indoor dataset. The dataset includes approximately 1.2 K images of common scenes within our lab environment. Notably, the environments created for testing differ from the ones used during the fine-tuning phase, since they incorporate new scenes and objects for evaluation. Since manual labeling is a very time-consuming activity, the Segment Anything Model (SAM) [26] is used to facilitate the mask creation process. SAM is designed to address the challenges of creating high-quality masks for various objects in images. It is trained on 11M images with over 1B masks and can produce valid segmentation masks in real-time, when prompted with different types of inputs such as points, boxes, and text. Once the masks are retrieved, we can assign the correct labels to each one of them, thus identifying the ground truth of each image. Furthermore, during training we employ a data augmentation pipeline to increase the dataset size. This pipeline is based on random resizing, random cropping, random flipping and photometric distortion. The fine-tuning parameters coincide with the one specified in Table I, with the exception that training continued for another 80 K iterations. This resumed from $LR = 10^{-4}$ and finished with $LR_{\min} = 1 \times 10^{-5}$.

The NN runs on-board the Jetson NX at 7.1 Hz and obtains a mean Intersection over Union (mIoU) of 67.51% and a mean Accuracy (mAcc) of 85.21%. For a qualitative evaluation of the segmentation results, please refer to Fig. 3.

C. Environment Assessment and Autonomous Landing

In our test scenario, we operate with a map resolution of 0.1 meters, while the designated safe landing zone measures 0.5×0.5 m^2. The acquisition of RGB images and the stereo pair occurs at a rate of 30 Hz, whereas the disparity maps runs at 3 Hz, which suffices for our operational speeds. The processed point clouds and the safe landing locations are instead published at a frequency of 1.1 Hz. However, if better performances are required, we have the flexibility to increase the updating frequency of the processed points. Finally, VIO runs at 100 Hz.

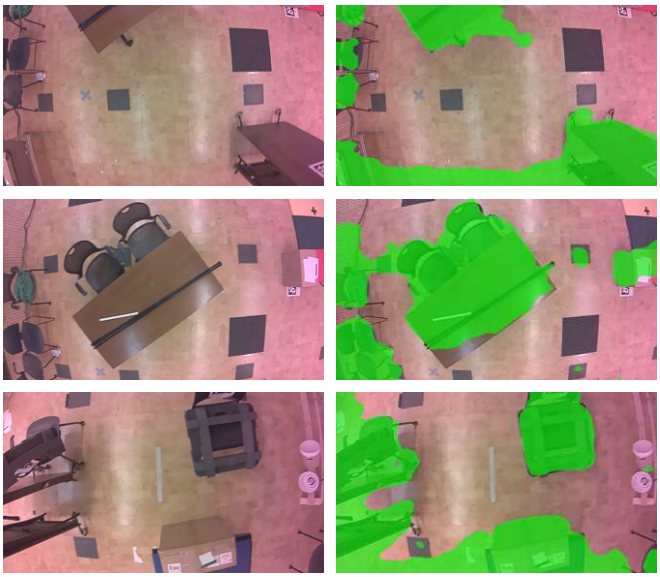


Fig. 3: Segmentation results in three different scenarios: on the *left* column the RGB images, on the *right* column the segmentation results. The green areas are considered unsafe.

For our tests, we select $\alpha = 0.65$ and $\beta = 0.35$. As detailed in eq. (5), we empirically observe that these settings prioritize the term related to the drone's proximity to the safe landing area over the distance between the safe landing site and obstacles. Moreover, the slope and roughness thresholds are set respectively to 15° and 0.05 m.

The proposed pipeline is tested in several scenarios, including different obstacle heights and densities and different navigation patterns and speeds. In Fig. 4, we showcase the mapping process and a full experiment in a low height, middle density obstacles environment. After take-off, the drone follows an "8" navigation pattern and performs a couple of flight runs over the environment. As we can see from the succession of images in Fig. 4(b), each time the drone perceives new areas, by following its path, the map is updated accordingly. When the navigation behaviour is concluded, the 2D occupancy grid is fully updated and the drone can implement the final environmental assessment. Once the safest landing area is identified, the drone finally implements the autonomous landing. The last picture in Fig. 4(b) shows the whole map with the safe landing zone location and the drone's path. Out of 7 tests performed in different challenging environments, the drone is able to safely land 6 times, showing its capability to detect a safe landing zone with an overall success rate of 85.71%. The unsuccessful landing is not attributed to any errors in segmentation or metric data but rather to the grid discretization. In this specific experiment, the grid size was excessively large in comparison to certain low-height obstacles. Through testing with a slightly smaller grid, we can effectively address

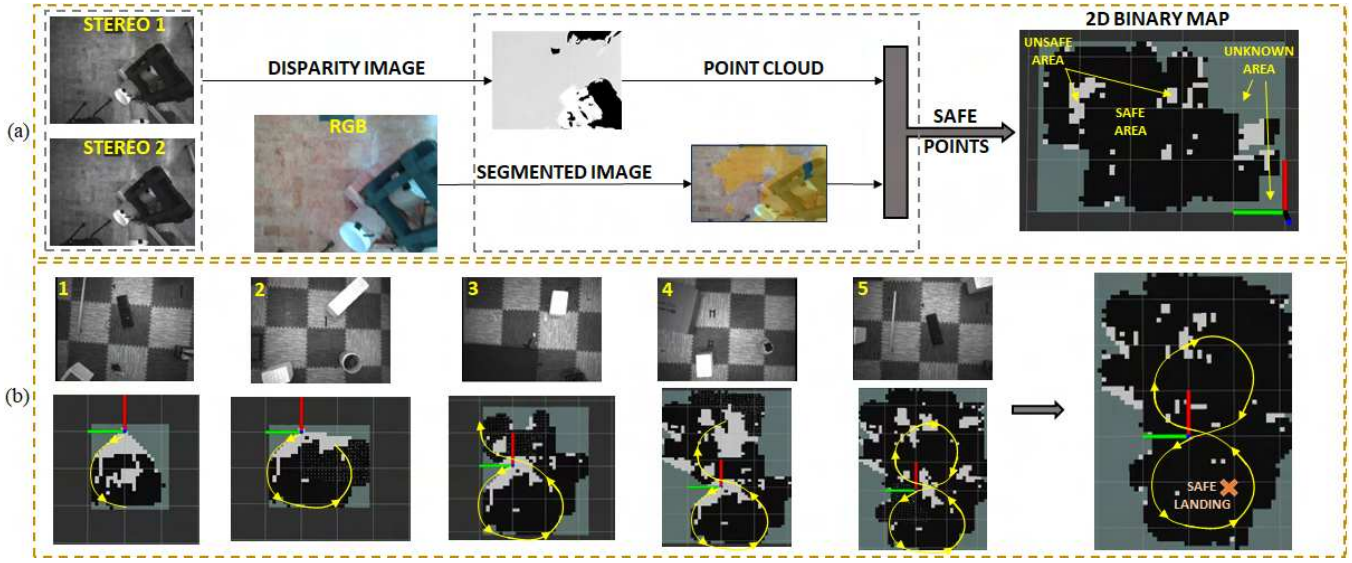
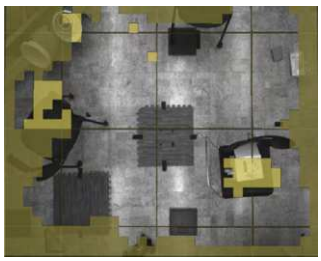


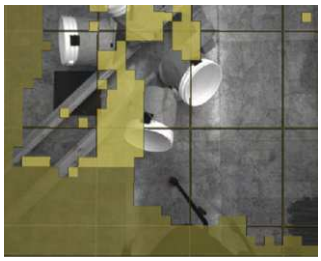
Fig. 4: *(a)* Data acquisition & processing pipeline for the map creation and *(b)* Site evaluation and safe autonomous landing experiment in a low height, middle density environment scenario with an "8" navigation pattern.

(a) High density, low height obstacles scenario. Acc = 76.5%.

(b) Long waypoints navigation pattern scenario. Acc = 82.6%.



(c) Medium density, medium height obstacle scenario 1. Acc = 91.4%.



(d) Medium density, medium height obstacle scenario 2. Acc = 75.1%.

Fig. 5: 2D binary map of safe and unsafe landing locations, overlayed to the real environment. The light green regions are unsafe while the dark green are unknown and still to be explored. Both of them are hazardous areas for landing.

this issue without impacting computational efficiency. By overlaying the created 2D occupancy grid with the top view of the environment, we can also quantitatively evaluate the number of zones correctly "classified" as safe or unsafe. We evaluate the classification accuracy as

$$\text{Acc} = \frac{TP + TN}{TP + FP + FN + TN}, \tag{6}$$

where TP are the true positive, TN the true negative, FP the false positive and FN the false negative. In Fig. 5, we show some of the results obtained during testing in four environments, considering different obstacles heights, densities, and navigation patterns. The mean accuracy (mAcc) in identifying safe landing zones is approximately 81.4%.

V. Conclusion

In this work, we have presented a visual approach to autonomously detect safe landing sites on-board a quadrotor, with limited SWaP resources. The proposed approach allows accurate and efficient safe landing detection since it combines both semantic and metric information and directly computes a 2D binary map of the overflown environment, thus avoiding the creation of expensive elevation maps. Furthermore, we have also shown its ability to guarantee real-time, safe autonomous landing in real-world environments.

In the future, we are considering to enhance the NN performances by adopting more sophisticated architectures, such as DeeplabV3+ [27]. In such a way we could improve the accuracy in identifying a safe landing zone, since this will in part be limited by the maximum network accuracy. However, optimization is needed to obtain similar inference times, given the higher model complexity. Moreover, we aim to enhance the point cloud processing and the mapping procedure. In particular, our current setup does not incorporate Simultaneous Localization and Mapping (SLAM) capabilities, therefore we are prone to drift over large scale environments. We are also thinking of using a monocular camera and learning-based, efficient depth estimation techniques for perceiving the environment, therefore avoiding the use of stereo cameras. Additionally, we would like to develop autonomous exploration environment strategies. This process can still leverage multiple cost metrics employed in this work. By prioritizing areas with lower costs, we can autonomously guide the drone to directly explore areas that appears to be safer, making our drone entirely self-sufficient.

Acknowledgment

We would like to thank Luca Morando and Pratyaksh P. Rao for their contributions and support to this project.

REFERENCES

[1] A. A. Laghari, A. K. Jumani, R. A. Laghari, and H. Nawaz, "Unmanned aerial vehicles: A review," *Cognitive Robotics*, vol. 3, pp. 8–22, 2023.

[2] A. Otto, N. Agatz, J. Campbell, B. Golden, and E. Pesch, "Optimization approaches for civil applications of unmanned aerial vehicles (uavs) or aerial drones: A survey," *Networks*, vol. 72, no. 4, pp. 411–458, 2018.

[3] G. Li, X. Liu, and G. Loianno, "Safety-aware human-robot collaborative transportation and manipulation with multiple mavs," 2023.

[4] M. Morita, H. Kinjo, S. Sato, T. Sulyyon, and T. Anezaki, "Autonomous flight drone for infrastructure (transmission line) inspection (3)," in *International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS)*, 2017, pp. 198–201.

[5] N. Gageik, P. Benz, and S. Montenegro, "Obstacle detection and collision avoidance for a uav with complementary low-cost sensors," *IEEE Access*, vol. 3, pp. 599–609, 2015.

[6] A. Devo, J. Mao, G. Costante, and G. Loianno, "Autonomous single-image drone exploration with deep reinforcement learning and mixed reality," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 5031–5038, 2022.

[7] C. S. Tang and L. P. Veelenturf, "The strategic role of logistics in the industry 4.0 era," *Transportation Research Part E: Logistics and Transportation Review*, vol. 129, pp. 1–11, 2019.

[8] H.-W. Choi, H.-J. Kim, S.-K. Kim, and W. S. Na, "An overview of drone applications in the construction industry," *Drones*, vol. 7, no. 8, 2023.

[9] C. Symeonidis, E. Kakaletsis, I. Mademlis, N. Nikolaidis, A. Tefas, and I. Pitas, "Vision-based uav safe landing exploiting lightweight deep neural networks," in *Proceedings of the 2021 4th International Conference on Image and Graphics Processing*, ser. ICIGP '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 13–19.

[10] P. Schoppmann, P. F. Proença, J. Delaune, M. Pantic, T. Hinzmann, L. Matthies, R. Siegwart, and R. Brockers, "Multi-resolution elevation mapping and safe landing site detection with applications to planetary rotorcraft," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 1990–1997.

[11] R. Moghe and R. Zanetti, "A deep learning approach to hazard detection for autonomous lunar landing," *The Journal of the Astronautical Sciences*, vol. 67, pp. 1–20, 10 2020.

[12] A. Johnson, J. Montgomery, and L. Matthies, "Vision guided landing of an autonomous helicopter in hazardous terrain," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2005, pp. 3966–3971.

[13] C. Forster, M. Faessler, F. Fontana, M. Werlberger, and D. Scaramuzza, "Continuous on-board monocular-vision-based elevation mapping applied to autonomous landing of micro aerial vehicles," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 111–118.

[14] K. Tomita, K. A. Skinner, and K. Ho, "Bayesian deep learning for segmentation for autonomous safe planetary landing," *Journal of Spacecraft and Rockets*, vol. 59, no. 6, pp. 1800–1808, 2022.

[15] X. Guo, S. Denman, C. Fookes, L. Mejias, and S. Sridharan, "Automatic uav forced landing site detection using machine learning," in *International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, 2014, pp. 1–7.

[16] C. Yu, C. Gao, J. Wang, G. Yu, C. Shen, and N. Sang, "Bisenet v2: Bilateral network with guided aggregation for real-time semantic segmentation," *International Journal of Computer Vision (IJCV)*, vol. 129, pp. 3051–3068, 2021.

[17] Nvidia tensorrt documentation. [Online]. Available: https://docs.nvidia.com/deeplearning/tensorrt/

[18] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, p. 381–395, jun 1981.

[19] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, 2013, software available at http://octomap.github.com. [Online]. Available: http://octomap.github.com

[20] J. Park, Y. Kim, and S. Kim, "Landing site searching and selection algorithm development using vision system and its application to quadrotor," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 2, pp. 488–503, 2015.

[21] G. Loianno, C. Brunner, G. McGrath, and V. Kumar, "Estimation, control, and planning for aggressive flight with a small quadrotor with a single camera and imu," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 404–411, April 2017.

[22] J. Mao, S. Nogar, C. M. Kroninger, and G. Loianno, "Robust active visual perching with quadrotors on inclined surfaces," *IEEE Transactions on Robotics*, vol. 39, no. 3, pp. 1836–1852, 2023.

[23] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: An open-source robot operating system," 2009.

[24] B. Zhou, H. Zhao, X. Puig, T. Xiao, S. Fidler, A. Barriuso, and A. Torralba, "Semantic understanding of scenes through the ade20k dataset," *International Journal of Computer Vision (IJCV)*, vol. 127, pp. 302–321, 2019.

[25] M. Contributors, "MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark," https://github.com/open-mmlab/mmsegmentation, 2020.

[26] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick, "Segment anything," 2023.

[27] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Computer Vision – ECCV 2018*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds. Cham: Springer International Publishing, 2018, pp. 833–851.