2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Light-weight approach for safe landing in populated areas

Tilemachos Mitroudas, Vasiliki Balaska, Athanasios Psomoulis and Antonios Gasteratos

***Abstract*—Landing safety is a challenge heavily engaging the research community recently, due to the increasing interest in applications availed by aerial vehicles. In this paper, we propose a landing safety pipeline based on state of the art object detectors and OctoMap. First, a point cloud of surface obstacles is generated, which is then inserted in an OctoMap. The unoccupied areas are identified, thus resulting to a sum of safe landing points. Due to the low processing time achieved by state of the art object detectors and the efficient point cloud manipulation using OctoMap, it is feasible for our approach to deploy on low-weight embedded systems. The proposed pipeline has been evaluated in many simulation scenarios, varying in people density, number, and movement. Simulations were executed with an Nvidia Jetson Nano in the loop to confirm the pipeline's performance and robustness in a low computing power hardware. The experiments yielded promising results with a 87% success rate.**

***Index Terms*—Safe Landing, Object Detectors, OctoMap, Point Cloud, PX4, UAV**

I. Introduction

Contemporary research in robotics evince that modern unmanned aerial vehicles (UAVs) are able to cope with a variety of diverging and demanding applications, such as terrain mapping, surveillance, search and rescue, photogrammetry, etc. [1], [2] Notwithstanding the increasing autonomy and intelligence endowed to UAVs, factors leading to their failure are still present [3]. Therefore, precaution is of paramount importance, especially in the case of a UAV overflying residential areas. During an event of a light system failure, such as loss of data-link, battery malfunction, etc., the most common fail-safe scenario is emergency landing. In such a situation, the ultimate goal would be to avoid injuring people in crowded regions [4]. Depending on the type of the aerial platform, e.g. fixed wing, multi-rotor, etc., and the flight scenario, e.g. indoor/outdoor, remote/populated area, safe landing might appear to be a complex problem, particularly in cases where a predefined fail-safe landing spot is proscribed [5]. Safe landing in crowded areas is a strenuous task due to the volatility of the environment, owed to multiple diverging parameters, such as moving people, variable terrain, camera moves, occlusions etc. Systems allowing safe touch down ideally need to operate in real-time and run locally on the UAV's on-board computer, on account of the fact that data-link loss constitutes a fairly usual failure [6]. Real-time performance is another trammel, as UAVs face more stringent limitations compared to other autonomous systems, owed to their physical limitations on weight, size, and battery capacity [7]. Recent advancements in deep learning methods led to algorithms efficient enough for real-time operation on low-end hardware [8], [9] [10], [11].

Motivated by the aforementioned barriers, in this paper we attempt to present a solution for safe UAV landing in human inhabited areas. The proposed method is meant to fit in embedded onboard hardware with the aim to facilitate the autonomy of the flying vehicle. The rest of the paper is organized as follows. Section II discusses the related work. Section III presents the proposed method. Section IV exhibits the experiments conducted to test the proposed pipeline, as well as the data produced. In Section V a useful discussion of our findings is provided, while in SectionVI conclusions are drawn and our plans for future work are presented.

II. Related work

The proposed work focuses on the extraction of safe landing zones (SLZ) through the detection of landing obstacles, in our case people. In general SLZ extraction algorithms aim to either detect flat regions or avoid landing obstacles such as vegetation, people, equipment, etc. The solution proposed by the authors in [12] is based on the concept that safe regions in images tend to have flat surfaces. Therefore, with the aim to extract those regions, Gabor filters have been employed. As a result, images with different orientations can be obtained and in each of them, the candidate pixels are grouped using Markov chain codes. The system detects the candidate region with the biggest area and then it estimates the degree of similarity between the reference region and the corresponding ones. In [13] the authors present a complete modular framework for landing site selection, which comprises a point cloud prepossessing module, a coarse landing site selection module, a fine terrain evaluation module and a landing optimal model. The pipeline was designed to achieve real-time performance. It was also tested in different real world known and unknown scenarios. A similar approach for fixed wing aircrafts was proposed in [14], where landing hazards are considered as terrain roughness and slope. The UAV's on-board extended Kalman filter considers the surrounding obstacles that occlude landing and calculates a landing approach path. A method based on a deep learning approach was presented in [5]. The proposed model was built on a semantic segmentation architecture. The thematic classes of the terrain are mapped into safety-scores for UAV landing.

Many contemporary landing obstacle detection algorithms, including those designed for crowd detection, leverage the efficiency and performance demonstrated by deep neural networks. The identification of obstacles at ground level is usually accomplished by estimating crowd densities. While useful, in the case of UAVs the constant change of perspective renders the image plane density with no immediate physical

979-8-3503-8457-4/24/$31.00 ©2024 IEEE

meaning. The authors in [15] model the scale changes and reason in terms of people per square-meter. Feeding this model to a network allows for scale consistency. In [16], the authors present a two-loss model where classification is supported by regression. This is performed by resorting to the construction of a spatial graph for each analyzed image and the evaluation of each cluster. The semantic information is derived by a class activation heat-map. Another crowd detection algorithm based on deep neural networks is presented in [17]. The authors' main contribution was the proposal of a lightweight architecture, in which a deep learning graph regularization technique is attached to all neural layers of their proposed CNN model. Another very interesting crowd detection method is suggested in [18] where the crowd density map is generated in each image and projected in the head plane. A binary occupancy map is generated and the SLZs are obtained as circular regions with a minimum security radius.

In the proposed approach, we put forward a modular framework[1] aiming to avoid landing obstacles such as people. The whole pipeline was designed to be as lightweight as possible so as to achieve real-time performance on lower-end hardware and thus be deployable on UAVs. The first module incorporates an object detector producing a point cloud, each point of which represents a detection. The second module aims to increase the performance of the system by inserting the point cloud into an OctoMap. By projecting each OctoMap node onto the ground plane we manage to create an occupancy grid containing only the landing-obstacles. Having outlined the operability of the proposed approach, we now can summarize its main contribution as follows:

- A modular lightweight system based on Robotic Operating System (ROS) [19], designed to be implemented in lower-end hardware, that capitalises on the advancements of the object detectors to find landing obstacles with great accuracy and low inference time.
- The representation of the landing obstacles, i.e. people, as OctoMap nodes, which boosts performance. Additionally, it copes with the uncertainty and inaccuracy of spatial measurements in a probabilistic manner. Finally, inserting the point cloud into an OctoMap not only frees up resources but additionally, it enables the map to be updated by different sensors adding a a semantic dimension to it.
- An SLZ detection algorithm that produces a redundant amount of landing point candidates. Hence, in case a spot is characterized occupied (due to changing environment), the closest unoccupied spot is available at any time.

III. Methodology

In this work, we put forward a complete lightweight system that builds an occupancy map of the terrain, chooses the safest landing point available and controls the overall landing procedure. The method relies upon two primary "pillars", namely state of the art object detectors and OctoMaps. Most of the recently reported works focus on DNN architectures, serving to achieve a safe landing on flat surfaces in the presence of moving obstacles, such as people. Our approach to the problem uses a state-of-the-art object detector, namely the YOLO V8, exploiting its real-time point cloud management is not very efficient and poses performance issues to low-power GPUs.Cloud Generator (PCG), the Temporal OctoMap (TOM) generator and the Landing Commander (LC). Our architecture is illustrated in Fig. 1. The communication between them is achieved through the exchange of predefined ROS messages, achieving true modularity in our system. In the rest of the Section, we shall extensively present each module comprising the proposed framework.

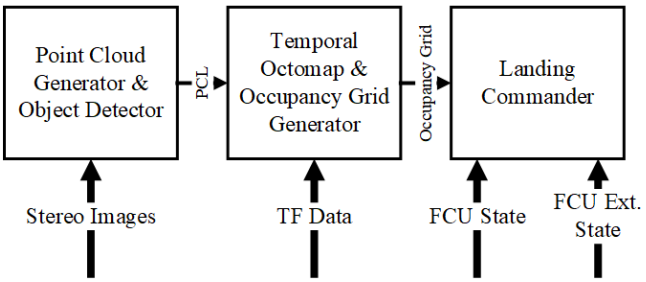

Fig. 1. The proposed framework consists of three sub-modules: The first one is based on the object detector; it receives a stereo pair from a camera and it outputs a point cloud, which is then fed into the second pipeline stage. The second module along with the point cloud receives transformed data necessary to register the point cloud onto an OctoMap.

A. Point Cloud Generator

Real-time object detection has concerned many computer vision scholars in their attempt to solve different practical problems [20]. A state of the art tool is the YOLO family of models. In particular, YOLO v8 consists a real time object detection and image segmentation [21]. Due to various optimizations the above model achieves high performance gains that strengthen the training for improving the accuracy without impacting its latency. The latter forms a key element to our pipeline, since it benefits from the consequent detections. The first pipeline module detects landing obstacles from a synchronised image pair acquired through a stereo camera mounted on the UAV. The detections in both images are matched with k-nearest neighbor [22] resulting in the position of the landing obstacle relatively to the camera frame. Finally, ROS "pointcloud2" message is populated with those observations. Note that, point cloud management is not very efficient and pose performance issues to low power GPUs.

B. Temporal OctoMap

An OctoMap is a 3D mapping representation of the environment, which provides an efficient probabilistic 3D mapping framework [23]. The primary purpose of the second pipeline module is to update the occupancy probabilities of the voxels in the octree by means of the point cloud data

[1]The pipeline along with the evaluation module can be found at GitHub: https://github.com/telemc97/safety_pipeline.git

generated from the previous module. Occupied voxels are projected onto the ground plane to form an occupancy grid, which comprises the output of this module. OctoMap handles measurements in a probabilistic manner to cope with errors generated from inertial and optical sensors. Due to the fact that mobile robots operate in a dynamic environment moving obstacles should be taken into consideration. Each Octree node is stamped and deleted on a specific timeline in case it is not re-initialized. To visualize that feature, each node is assigned with a color dependent on its remaining lifetime. Red signifies newly initialized nodes, while blue ones are close to their lifetime end as seen in Fig. 2. Another issue of embedded systems is their lack of processing power. Thus, algorithms ought to be as efficient as possible with respect to memory access time and consumption. To facilitate the communication between the modules and further comply with the goal of modularity, the occupancy data is transmitted to the last module using ROS occupancy grid message.

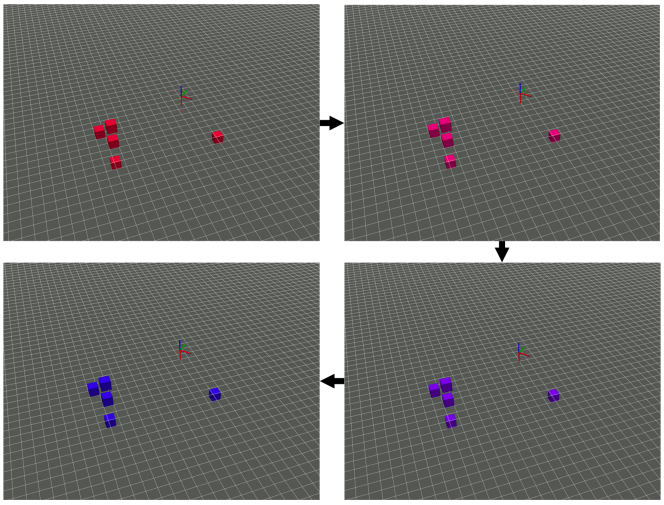

Fig. 2. Temporal Visualization: Node Lifetimes and Time-Dependent Dynamics in Detection Systems-Each colored node represents the lifetime of each detection. In the case that, the lifetime of each node is t_{max} then at time $t = t_{max}$ the node would be red, while at $t = 0$ it would be blue. A node that has reached its lifetime will be deleted unless reinitialized. The arrows depict the succession of continuous snapshots of the visualization, emphasizing the color change of the old detections.

C. Landing Commander

The Occupancy Grid is ultimately fed to the last component of this system. The Landing Commander aims to extract safe landing zones from the Grid and store them efficiently in an k-d tree implementation. It is also responsible for the landing initiation upon arriving at the closest unoccupied point. In order to extract the closest landing points to the UAV, we search the grid with a process closely resembling a discrete convolutional function.

$$g(x, y) = \kappa * f(x, y), \tag{1}$$

$$\kappa = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}. \tag{2}$$

By applying this, we create a "heat map" where each cell with value $0 < x < 1$ denotes grids close to an obstacle (safety radius) and $x = 1$ indicates the existence of an obstacle.

Once more, the process followed to find the unoccupied cells closely resamples a convolution, using an identity kernel. In case that $g(x, y) = 0$ the cell is unoccupied, the euclidean distance from the UAV is calculated and along with its coordinates (x, y), is inserted in a k-d tree.

For efficiency purposes, the inspection of the occupancy grid is required to be dynamic. Thus, the stride is variable and proportional to the processing time of previous iterations. The processing time (t) is dependent on parameters of the occupancy grid, such as its size $(h \times w)$, the ratio between occupied and unoccupied cells (r), the stride (s) and the kernel size (b^2). It is approximately calculated by the Eq. 3:

$$t = \left(\frac{hw(1-r)}{s^2}\right) b^2. \tag{3}$$

Solving for stride (s) and multiplying with an intensity coefficient (a) results in:

$$s = \left(\sqrt{\frac{b^2 hw - hwr}{t}}\right) a. \tag{4}$$

By incorporating Eq. 4 in our algorithm, we can keep the processing time of the function at a steady rate, regardless of the occupancy grid size (Fig. 3). The adjustment of the a coefficient is contingent on the system's performance. Lowering the a will result in fewer probable landing spots but also in quicker processing times. In that way, we are able to prioritize a low execution times over the amount of landing spots, mainly in larger maps.

IV. Experimental Results

A. Experimental Process

This section presents the experiments performed to evaluate our system. As mentioned in Section III, the pipeline consists of three modules, each of which necessitates to be assessed independently while the whole system also needs to be evaluated as well. The processing time of each module is logged along with the training metrics of the network used in the PCG.

Our pipeline has been extensively tested in the Gazebo environment. The UAV autopilot of our choice is PX4 [24] due to ROS compatibility. The ease in take-off and landing that the quadrotor layout exhibits, renders it the most suitable for our application [25], [26]. The model used in the Gazebo simulation was the one presented in [27], [28].

A dynamic test environment was designed populated with both static and moving people. The behaviour of the latter is

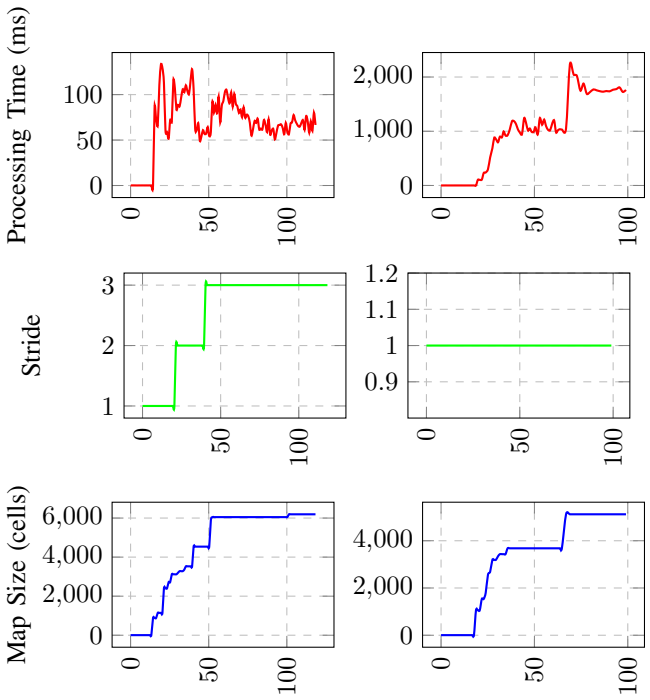

Fig. 3. The performance metrics of the pipeline are presented in these diagrams. In the left column, the dynamic stride is incorporated in the matrix checking, while in the right column, the stride is set at the fixed value of 1. Note that, the dynamic stride achieves approximately constant processing time of $70ms$ regardless of the matrix size.

based on the Social Force Model for Social Local Navigation [29], [30].

TABLE I
HARDWARE SETUPS THAT EXPERIMENTS WERE CARRIED ON

Setup	CPU	GPU	RAM
Setup #1	Intel i7-9750H	RTX 2060	16Gb
Setup #2	Quad-core ARM Cortex-A57	NVIDIA Maxwell architecture (128 CUDA cores)	4Gb

In total, 8 experiments were conducted (Table. III). More specifically, experiments #0-#3 were performed in setup #1 using yolov8s (#0, #1) and yolov8n (#2, #3). Experiments #4-#7 were performed with setup #2 in the loop utilizing once again yolov8s and yolov8n. In that way, we are able to confirm that the system can be deployed effectively in less capable hardware.

An independent module was created to evaluate the performance and accuracy of the entire system. This is achieved by reconstructing a ground truth occupancy map from the simulated Gazebo world (A) and comparing it with the occupancy grid produced by the third module of our pipeline (B). As a result, their IoU is derived:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}. \tag{5}$$

The cumulative total of landing spots, including those appropriately identified as unoccupied, is also recorded as performance metrics. Lastly, upon a successful landing, the Euclidean distance to the nearest landing obstacle is also logged as proposed in [7].

B. DNN Training & Dataset

As mentioned in the previous section, YOLO v8 is a state of the art object detector that constitutes a pillar of our pipeline. We trained our network on the Okutama Action Dataset, which consists of bird's-eye view images of people captured by UAVs flying at various altitudes and angles. Three models of the YOLO v8 architecture were trained and used in our experiments, more specifically, yolov8m, yolov8s and yolov8n. Their training metrics are shown in Table II.

TABLE II
TRAINING RESULTS ON OKUTAMA ACTION DATASET

Model	Precision	$\text{mAP}_{0.5}$	$\text{mAP}_{0.5:0.95}$	Epochs
yolov8m	0.8240	0.80794	0.29752	8
yolov8s	0.8005	0.7730	0.28337	13
yolov8n	0.8099	0.7775	0.28097	23

C. Processing times

One of the primary objectives of this pipeline was to achieve real-time performance across every module. While the processing time of the first two modules is hardware depended, The execution rate of the LC depends on scenario-specific parameters, such as mission area, landing obstacles, etc. As seen in Fig. 3, the dynamic stride manages to keep the processing time of the LC module stable at an average of $30ms$ in the case of Setup #1 and $60ms$ in the case of the Setup #2 (Nvidia Jetson Nano). Based on the adjustments in the stride coefficient (a in Eq. 4), we can increase the number of possible landing points by sacrificing the performance. Therefore, a value must be defined based on the performance of each system and the target latency.

We conclude that the only module with a significant deviation in performance is the PCG which is GPU dependent, By using lighter models such as yolov8n this can be combated achieving greater performance.

D. Overall Metrics

The entire pipeline is evaluated by the percentage of landing points rightfully characterized as unoccupied. Moreover upon a successful landing, the euclidean distance between the UAV and the closest landing obstacle is also logged as a performance metric The algorithm excludes areas in a certain radius for each detected landing obstacle. In our tests, the radius was set to 2m hence, in case that the shortest euclidean distance is less than 2m, the test was determined as failed. Ultimately from the 8 performed tests, all of them apart from one managed to successfully land at a distance equal or greater to the minimum radius from a landing obstacle. The failure of experiment #6 is attributed to the constant movement of the people on the simulation environment. On the other hand, the satisfactory results from test #5 (Fig. 5) are attributed to the greater length of the mission and consequently also to the greater number of observations.

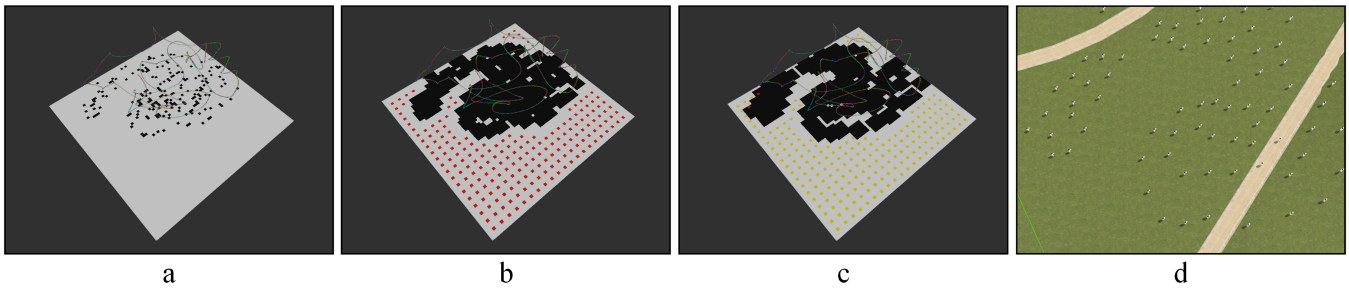

Fig. 4. This is a representative visualization of the output of each sub-module along with the aerial picture of the densely populated environment. Shown in **a** is the occupancy mesh produced by the TOM module after importing the Point Cloud into OctoMap. In **b**, the safety margins are applied to each landing obstacle. The available landing points are presented in red, while the active landing point is highlighted in green. In Fig. 4**c**, the output of the evaluation module is shown. Every correct land point (not Occupied) is presented with yellow, while every landing point that has been mistakenly considered unoccupied is presented with red. The actual simulated world (terrain, people, road) created on Gazebo is depicted on **d**.

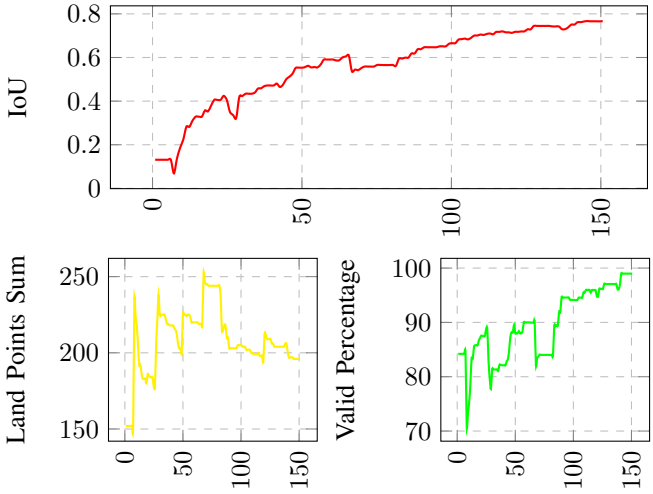

Fig. 5. The IoU increases as the UAV hovers over the area. The proportion of the land points that are indeed unoccupied is growing along with the mission eventually reaching 100% accuracy.

TABLE III
SOME INDICATIVE RESULTS OF THE VARIOUS EXPERIMENT SCENARIOS

Test #.	Success	Min. Dist.	Setup # & model	IoU	Valid Points
0	True	3m	#1, s	0.62	99.7%
1	True	2m	#1, s	0.55	96.9%
2	True	2m	#1, n	0.68	97.8%
3	True	3m	#1, n	0.69	97.2%
4	True	3m	#2, s	0.69	98.1%
5	True	2m	#2, s	0.78	100%
6	False	2m	#2, n	0.6	97.9%
7	True	2m	#2, n	0.7	100%

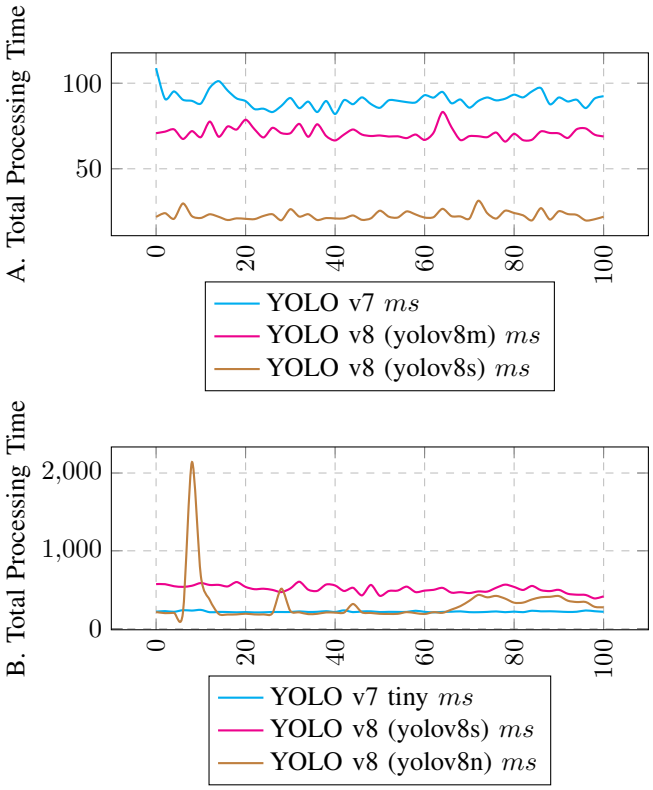

Fig. 6. YOLO v8 forms a very robust object detector with excellent performance and accuracy. The pipeline is positively affected by it since its effectiveness depends on the consequent detections. Setup #1 is used in A and Setup #2 in B.

V. Discussion

The objective of the proposed pipeline is to be modular, lightweight, and easily deployable on different platforms. To achieve modularity, we decomposed the problem into three sub-modules, each with discrete input and output as seen in Fig. 6. The efficiency of our system is mainly attributed to the fact that the spatial information is stored as an OctoMap. The ease of deployment in a variety of platforms was achieved by developing the whole pipeline using ROS [31]. The pipeline in the context of this paper is limited in detecting and avoiding people during landing at a given moment; by exploiting the tracking capabilities of the YOLO v8 framework, we could track and predict the position of an obstacle, thus avoiding dangerous scenarios with greater accuracy. Due to modularity, the system can be extended in order to assign each landing spot a score based on a multi-criteria analysis in relation to slope, people density and activity, distance from base or UAV, detection confidence

etc. Ultimately in future work we plan to capitalize on the tracking capabilities of the YOLO v8 framework to track and predict the position of landing obstacles avoiding scenarios similar to experiment #6.

VI. Conclusion

To conclude we propose a landing safety pipeline which is based on state of the art object detectors and OctoMap. It consists of three modules: The first one creates a point cloud which comprises the object detector's detection in world coordinates, the second module converts this point cloud to OctoMap to generate an occupancy grid projecting the occupied areas on the ground plane, and the last module takes on the detection of landing points and oversees the landing procedure upon reaching them. Achieving real time performance is of utmost priority for our solution while, the ease of deployment in a variety of platforms will facilitate its further development. The pipeline can be improved by introducing a decision tree procedure in the landing module to determine the best landing point according to certain conditions (e.g. remaining battery, RSSI signal strength, etc.).

Acknowledgement

This research has been co-financed by the Hellenic Aerospace Industry-project UAV DESIGN AND DEVELOPMENT code: KE83026), as also by the European Health and Digital Executive Agency (HADEA), under the powers delegated by the European Commission ('European Commission') (project code: MASTERMINE-101091895).

References

[1] B. Alzahrani, O. S. Oubbati, A. Barnawi, M. Atiquzzaman, and D. Alghazzawi, "Uav assistance paradigm: State-of-the-art in applications and challenges," *Journal of Network and Computer Applications*, vol. 166, p. 102706, 2020.

[2] I. T. Papapetros, V. Balaska, and A. Gasteratos, "Multi-layer map: Augmenting semantic visual memory," in *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2020, pp. 1206–1212.

[3] S. Charalampidou, E. Lygouras, I. Dokas, A. Gasteratos, and A. Zacharopoulou, "A sociotechnical approach to uav safety for search and rescue missions," in *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2020, pp. 1416–1424.

[4] J. A. Gonzalez-Trejo and D. A. Mercado-Ravell, "Lightweight density map architecture for uavs safe landing in crowded areas," *J. of Intell. & Robot. Syst.*, vol. 102, no. 1, pp. 1–15, 2021.

[5] S. Abdollahzadeh, P.-L. Proulx, M. S. Allili, and J.-F. Lapointe, "Safe landing zones detection for uavs using deep regression," 2022, pp. 213–218.

[6] M. Shah Alam and J. Oluoch, "A survey of safe landing zone detection techniques for autonomous unmanned aerial vehicles (uavs)," *Expert Syst. with Appl.*, vol. 179, p. 115091, 2021.

[7] C. Symeonidis, E. Kakaletsis, I. Mademlis, N. Nikolaidis, A. Tefas, and I. Pitas, "Vision-based uav safe landing exploiting lightweight deep neural networks," 2021, pp. 13—19.

[8] E. Lygouras, A. Gasteratos, K. Tarchanidis, and A. Mitropoulos, "Rolfer: A fully autonomous aerial rescue support system," *Microprocessors and Microsystems*, vol. 61, pp. 32–42, 2018.

[9] E. Lygouras, N. Santavas, A. Taitzoglou, K. Tarchanidis, A. Mitropoulos, and A. Gasteratos, "Unsupervised human detection with an embedded vision system on a fully autonomous uav for search and rescue operations," *Sensors*, vol. 19, no. 16, p. 3542, 2019.

[10] J.-W. Lee, W. Lee, and K.-D. Kim, "An algorithm for local dynamic map generation for safe uav navigation," *Drones*, vol. 5, no. 3, 2021.

[11] E. Kakaletsis, C. Symeonidis, M. Tzelepi, I. Mademlis, A. Tefas, N. Nikolaidis, and I. Pitas, "Computer vision for autonomous uav flight safety: An overview and a vision-based safe landing pipeline example," no. 9, 2021.

[12] M. A. Kaljahi, P. Shivakumara, M. Y. I. Idris, M. H. Anisi, T. Lu, M. Blumenstein, and N. M. Noor, "An automatic zone detection system for safe landing of uavs," *Expert Syst. with Appl.*, vol. 122, pp. 319–333, 2019.

[13] L. Yang, C. Wang, and L. Wang, "Autonomous uavs landing site selection from point cloud in unknown environments," *ISA Trans.*, vol. 130, pp. 610–628, 2022.

[14] T. Hinzmann, T. Stastny, C. Cadena, R. Siegwart, and I. Gilitschenski, "Free lsd: Prior-free visual landing site detection for autonomous planes," *IEEE Robot. and Automat. Letters*, vol. 3, no. 3, pp. 2545–2552, 2018.

[15] W. Liu, K. Lis, M. Salzmann, and P. Fua, "Geometric and physical constraints for drone-based head plane crowd density estimation," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 244–249.

[16] G. Castellano, C. Castiello, C. Mencar, and G. Vessio, "Crowd detection in aerial images using spatial graphs and fully-convolutional neural networks," *IEEE Access*, vol. 8, pp. 64 534–64 544, 2020.

[17] M. Tzelepi and A. Tefas, "Graph embedded convolutional neural networks in human crowd detection for drone flight safety," *IEEE Trans. on Emerg. Topics in Comput. Intell.*, vol. 5, no. 2, pp. 191–204, 2021.

[18] J. González-Trejo, D. Mercado-Ravell, I. Becerra, and R. Murrieta-Cid, "On the visual-based safe landing of uavs in populated areas: A crucial aspect for urban deployment," *IEEE Robot. and Automat. Letters*, vol. 6, no. 4, pp. 7901–7908, 2021.

[19] R. Mehrotra, K. Namuduri, and N. Ranganathan, "Gabor filter-based edge detection," *Pattern Recognition*, vol. 25, no. 12, pp. 1479–1494, 1992.

[20] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," 2022.

[21] G. Jocher, A. Chaurasia, and J. Qiu, "YOLO by Ultralytics," 1 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[22] S. Leutenegger, M. Chli, and R. Y. Siegwart, "Brisk: Binary robust invariant scalable keypoints," in *2011 International Conference on Computer Vision*, 2011, pp. 2548–2555.

[23] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Auton Robots*, vol. 34, no. 3, pp. 189–206, 2013.

[24] L. Meier, D. Honegger, and M. Pollefeys, "Px4: A node-based multithreaded open source robotics framework for deeply embedded platforms," in *2015 IEEE Internatio Conference on Robotics and Automation (ICRA)*, 2015, pp. 6235–6240.

[25] W. Kwon, J. H. Park, M. Lee, J. Her, S.-H. Kim, and J.-W. Seo, "Robust autonomous navigation of unmanned aerial vehicles (uavs) for warehouses' inventory application," *IEEE Robot. and Automat. Letters*, vol. 5, no. 1, pp. 243–249, 2020.

[26] V. Balaska, D. Folinas, F. K. Konstantinidis, and A. Gasteratos, "Smart counting of unboxed stocks in the warehouse 4.0 ecosystem," in *2022 IEEE International Conference on Imaging Systems and Techniques (IST)*, 2022, pp. 1–6.

[27] T. Mitroudas, K. A. Tsintotas, N. Santavas, A. Psomoulis, and A. Gasteratos, "Towards 3d printed modular unmanned aerial vehicle development: The landing safety paradigm," 2022, pp. 1–6.

[28] T. Mitroudas, V. Balaska, A. Psomoulis, and A. Gasteratos, "Multi-criteria decision making for autonomous uav landing," in *2023 IEEE International Conference on Imaging Systems and Techniques (IST)*, 2023, pp. 1–5.

[29] M. Moussaïd, D. Helbing, S. Garnier, A. Johansson, M. Combe, and G. Theraulaz, "Experimental study of the behavioural mechanisms underlying self-organization in human crowds," *Proc. of the Royal Society B: Biol. Sciences*, vol. 276, no. 1668, pp. 2755–2762, 2009.

[30] M. Moussaïd, N. Perozo, S. Garnier, D. Helbing, and G. Theraulaz, "The walking behaviour of pedestrian social groups and its impact on crowd dynamics," *PLOS ONE*, vol. 5, pp. 1–7, 04 2010.

[31] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng *et al.*, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Softw.*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.