frontiers | Frontiers in Robotics and AI

TYPE Original Research
PUBLISHED 23 November 2023
DOI 10.3389/frobt.2023.1223157

OPEN ACCESS

EDITED BY
Patrick Sebastian,
University of Technology Petronas, Malaysia

REVIEWED BY
Minhyeok Lee,
Chung-Ang University, Republic of Korea
Haijie Zhang,
Optum, United States

*CORRESPONDENCE
Phuoc Thuan Nguyen,
tpnguy@utu.fi

RECEIVED 15 May 2023
ACCEPTED 01 November 2023
PUBLISHED 23 November 2023

CITATION
Nguyen PT, Westerlund T and Peña Queralta J (2023), Vision-based safe autonomous UAV docking with panoramic sensors.
Front. Robot. AI 10:1223157.
doi: 10.3389/frobt.2023.1223157

Vision-based safe autonomous UAV docking with panoramic sensors

Phuoc Thuan Nguyen[1,2]*, Tomi Westerlund[1] and Jorge Peña Queralta[1]

[1]Turku Intelligent Embedded and Robotic Systems (TIERS) Lab, University of Turku, Turku, Finland, [2]Computing Sciences, Faculty of Information Technology and Communication Sciences, Tampere University, Tampere, Finland

The remarkable growth of unmanned aerial vehicles (UAVs) has also sparked concerns about safety measures during their missions. To advance towards safer autonomous aerial robots, this work presents a vision-based solution to ensuring safe autonomous UAV landings with minimal infrastructure. During docking maneuvers, UAVs pose a hazard to people in the vicinity. In this paper, we propose the use of a single omnidirectional panoramic camera pointing upwards from a landing pad to detect and estimate the position of people around the landing area. The images are processed in real-time in an embedded computer, which communicates with the onboard computer of approaching UAVs to transition between landing, hovering or emergency landing states. While landing, the ground camera also aids in finding an optimal position, which can be required in case of low-battery or when hovering is no longer possible. We use a YOLOv7-based object detection model and a XGBooxt model for localizing nearby people, and the open-source ROS and PX4 frameworks for communication, interfacing, and control of the UAV. We present both simulation and real-world indoor experimental results to show the efficiency of our methods.

KEYWORDS

unmanned aerial vehicle (UAV), safe landing, deep learning, object detection, panoramic camera, vision-based localization

1 Introduction

Recently, unmanned aerial vehicles (UAVs, or drones) have seen an unprecedented rise in their adoption rate, primarily thanks to technological advancements improving their availability and dependability (Nex et al., 2022). They have been vital components in multiple civil applications, ranging from remote sensing applications (Xiang and Tian, 2011) to aerial delivery (Song et al., 2018).

One of the key issues stopping wider adoption of UAVs for civilian applications in urban areas is safety and security (Milano et al., 2022). Autonomous UAVs flying over populated areas pose inherent hazards. The risk increases significantly during take-off and docking maneuvers, with potential risks for nearby passers. This paper seeks to address the safety of persons near a landing area and define a framework for safety-aware autonomous UAV landing with minimal ground infrastructure.

Specifically, the aim is to first design and develop a solution with minimal infrastructure footprint and commercial off-the-self components. Then, we validate the functionality of

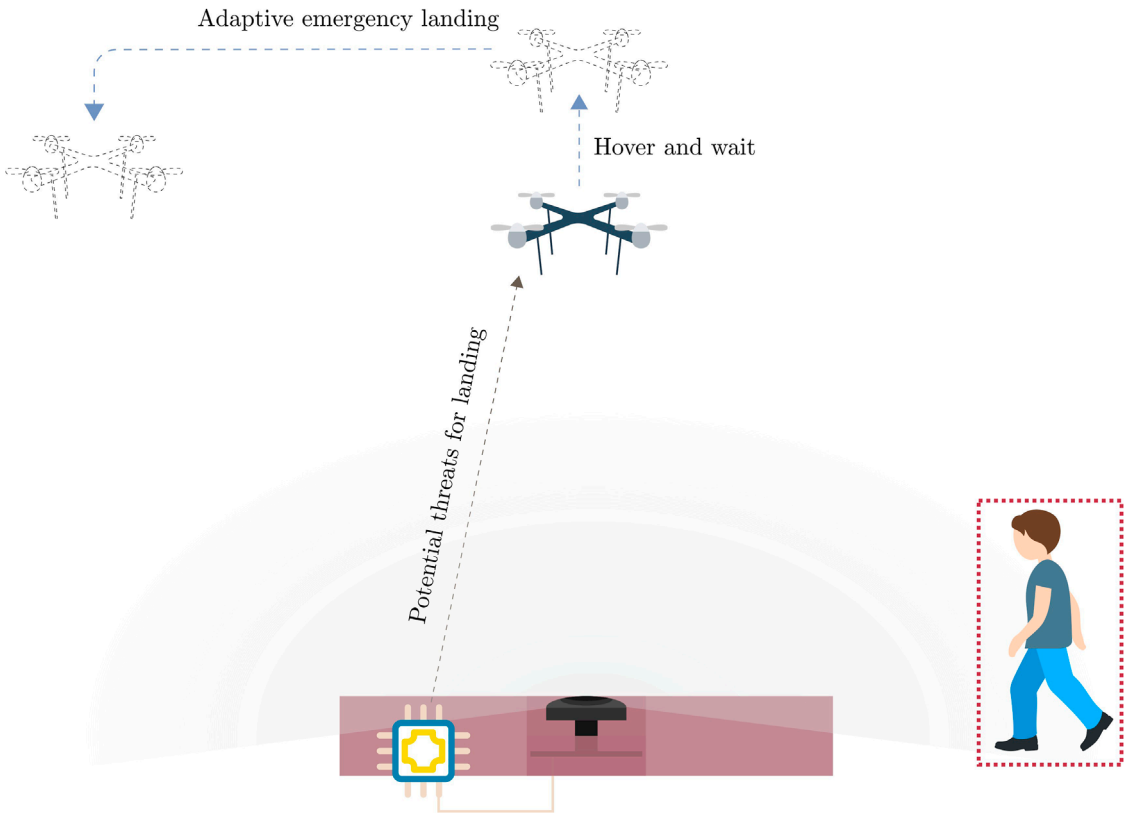

FIGURE 1
Envisioned system.

the system through a series of experiments in the Gazebo simulator and our $9 \times 8 \times 5$ m indoor test area. Our goal is to provide a solution that can further enhance the safety of autonomous UAV landing operations. One of the fundamental aspects of UAV landing safety is the avoidance of potential hazards on the landing path. While the concept of hazards avoidance during UAV landing is vast, we narrow the scope to protecting pedestrians near the landing area.

Leveraging on the recent rapid development of deep-learning-enabled computer vision on embedded hardware (Bhowmik and Appiah, 2018) and the high potential of 360° panoramic sensors, we approach the problem with an on-ground vision-based system for landing area monitoring to identify people who are at risk from the landing UAV. Our envisioned system is a lightweight landing pad with a single panoramic camera in the center providing a bottom-view that gives the system a 360° view of the surroundings. An embedded computing unit processes the information to generate relevant information, and packages them as lightweight, efficient messages to send to the UAV to adjust its landing trajectory. Figure 1 illustrates our envisioned system and the intended behavior.

On-ground approach for safe UAV landing have two significant advantages over its onboard counterpart. First, it widens the options for computing platforms and sensors. UAV payloads are limited, so for tasks such as aerial delivery, every gram of weight that can be saved by replacing heavy companion computers and sensors with more lightweight options is directly transferred to the weight that their primary task requires them to carry. The solution described and implemented in this paper does not involve a very high-end computing platform. Second, ground-based solutions are potentially more robust to limited environment observability from UAVs, and can also serve as a redundant way of ensuring safety in such critical scenarios.

Moreover, we design and implement the safe UAV landing software based on open-source libraries. Our software components include the detection module, which consists of an object detector and a distance estimator to identify and localize people in a two-dimensional space, and an autonomous flight program that safely allows the UAV to land while maintaining complete autonomy by using the information about the surroundings provided by the detection module. The functionality of each software component and the communication between them is facilitated by the free and open-source Robot Operating System (ROS), which has become the *de facto* standard for robotic applications in recent years. The popular autopilot library PX4 is also utilized for high-level UAV control and integration of autonomous flight algorithms.

The rest of this paper is organized as follows. Section 2 introduces related works in computer vision for panoramic sensors, and vision-based approaches used in UAV landing. Section 3 introduces our methodology for a ground-based vision-based safe

UAV landing framework. Section 4 then reports our experimental setup and results. Finally, Section 5 concludes the work.

2 Related works

2.1 Vision-based systems for autonomous UAV landing

In the literature, research for vision-enabled autonomous landing systems for UAVs, primarily multi-rotor vehicles, can be divided into two main categories: onboard and on-ground. According to the survey by Kong et al. (2014), the former approach is the more predominant and well-studied approach, with multiple systems developed for landing on known, unknown and moving areas, while works done for on-ground vision-based landing systems are still scarce. Most of the research to date presenting vision-based control approaches assume that vision sensors are located on the UAV (Kendoul, 2014; Ho et al., 2017; Zhang and Zhao, 2017).

A common point among on-ground systems is that they utilize a diverse range of sensing units because these systems are not restricted by UAV payload. However, most of the work in this category focuses on the pose estimation and control of the UAV rather than the monitoring of the landing site. In one of the earlier research on on-ground monitoring systems, Wang et al. (2006) introduced a computer control camera platform to identify square markers with known size patched on micro aircraft to measure their three-dimensional coordinates. The main limitation of this method was the camera's narrow field of view (FOV) and reliance on a step motor to shift its orientation to access other viewpoints. Martínez et al. (2009) later introduced a system that can estimate UAV's position based on onboard key features in real-time by extracting information provided by a trinocular camera system on the ground. Alternatively, instead of standard RGB cameras, Yang et al. (2016) presented a ground-based guidance system utilizing an array of near-infrared cameras, which significantly increases the detection range to detect, track, and autonomously land a fixed-wing UAV without reliance on GPS data. Other ground-based systems for detecting and tracking of UAVs based on 3D lidars have also been recently introduced (Catalano et al., 2023a; Catalano et al., 2023b; Sier et al., 2023).

Several other works have also presented onboard methods that select safe landing zones by detecting potential hazards on the landing path (Alam and Oluoch, 2021). In these papers, the authors utilize lightweight convolutional neural networks such as YOLO (Safadinho et al., 2020) and MobileNet (Castellano et al., 2020) to detect safe landing zones, which are away from individual or groups of people in populated areas (Tovanche-Picon et al., 2022), or flat and obstacle-free areas (Marcu et al., 2018).

Overall, most on-ground approaches for safe UAV docking focus on the detection and localization of incoming vehicles and are reliant on large compositions of sensing units. On the other hand, current onboard methods emphasize safe landing area selection by exploiting the large FOV from the UAVs' perspective. Additionally, thanks to the rapid development of embedded hardware and more efficient algorithms such as MobileNet (Howard et al., 2017), deep learning models have been the method of choice for safe landing systems. Nevertheless, we have found no previous works utilizing a panoramic sensor as the primary sensing unit. An advantage of 360° panoramic sensors that we aim to exploit is the wide FOV that they provide and their compact footprints (e.g., the dimensions of the single-lens PICAM360 module used in our work is $85 \times 56 \times 50$ mm.

2.2 Object detection on panoramic images

Object detection on panoramic images is a topic that is also less well-studied than its pinhole counterpart within the literature. One concept that has been researched to adapt object detection models to fisheye imagery, which frequently has oriented and radially distorted objects, is alternative representations for standard bounding boxes. Rashed et al. (2021) explored the usage of curved boxes, oriented boxes, ellipses, and polygons. YOLOv3 (Redmon and Farhadi, 2018) was adapted and modified to output these different representations. The results show that 24-sided polygons achieved the most reasonable tradeoffs between model complexity and accuracy. Further analysis also reports no drops in inference speed when increasing the number of vertices. Alternatively, (Xu et al., 2021), proposed a simple framework for oriented box representation by gliding each vertex of the original horizontal box on its corresponding side to get more accurate coverage of the detected object and demonstrated the method's effectiveness in object detection on aerial images, texts, and pedestrians in fisheye images.

Zhu et al. (2019) presented a localization method by leveraging top-view fisheye images from a UAV and altitude data. The proposed framework first involves acquiring pixel positions of objects using an object detection model implemented based on the RetinaNet model (Lin et al., 2017) with MobileNet (Howard et al., 2017) backbone for more efficient computing. Then, by fusing the camera's parameter and height data from other sensors, a series of coordinate transforms is performed to obtain the object's position in world coordinates.

In addition, some public fisheye image datasets were published to facilitate the development of this field of research. Most noticeable is the Woodscape dataset (Yogamani et al., 2019) for autonomous driving, comprising over 100,000 images from four surrounding cameras. Later, in 2022, the KITTI-360 dataset (Liao et al., 2022) was released as a successor to the popular KITTI dataset (Geiger et al., 2012). It expanded on the original work with more data for suburban driving from multiple sensor units, including two 180° fisheye cameras on each side of the station wagon.

3 Methodology

3.1 Detection module

To identify whether the landing spot is safe, we implement a system that detects people around the area and estimates their distances to the camera. For the rest of this work, we will refer to this combination of human detection and distance estimation system as the *detection module*.

3.1.1 Human detection

The human detection approach must operate with tight latency bounds while maintaining satisfactory accuracy to ensure safety for UAV landing operations. Since its inception, YOLO has been the *de facto* standard for real-time object detection (Redmon et al., 2016; Zou et al., 2019). Thanks to its unified network structure, the training process is end-to-end, significantly simplifying transferring the model to other datasets. Furthermore, the model makes a small sacrifice in prediction accuracy but significantly improves computational speed thanks to the low overhead in the detection phase. Previous successes in robotic vision have substantiated YOLO's real-time detection capability (Qingqing et al., 2020; Safadinho et al., 2020). This project's baseline object detection model is YOLOv7, the latest culmination of the YOLO series of object detection by the time of writing this paper. From the reported results, it has significantly outperformed its previous iterations regarding inference speed and accuracy. For more information about the details of YOLOv7, we refer the reader to the original work (Wang et al., 2022).

The official YOLOv7 project provides different model versions with varying sizes and complexity. The standard models are the tiny version, which optimizes for high throughput and minimal footprint to run on edge GPU; the normal version, namely YOLOv7, for regular consumer-grade GPUs; and the more powerful, cloud GPU-oriented YOLOv7-W6. To further optimize YOLOv7-tiny for edge GPUs, the authors use Rectified Linear Unit (ReLU) as the activation function. On the other hand, for other versions, Sigmoid-weighted Linear Unit (SiLU) (Elfwing et al., 2018) is used as the activation function. For this work, we mainly consider the tiny and the normal versions of YOLOv7, as empirical testing shows they are more suitable for deployment on our embedded platform.

3.1.2 Distance estimation

Monocular depth estimation is a challenging topic that has received much attention recently. In our envisioned system, the camera is placed in a stationary position. Therefore, algorithms that rely on the camera's motion (Ho et al., 2017) are unsuitable for our implementation. The most common approach for stationary monocular depth estimation is to train a deep learning model to predict depth from an arbitrary input image (Ranftl et al., 2020; Zhao et al., 2020). The training data can be from multiple measuring tools like LIDAR, RGB-D, and stereo cameras. Unfortunately, most public datasets only have depth images in perspective view. To our knowledge, no available pre-trained monocular depth estimation models trained on data with the same characteristics as ours exist. Another possible approach that has been studied is integrating a distance estimator head into the object detector's architecture (Vajgl et al., 2022).

The goal for the system is not to prioritize precisely predicting the distance of the person to the camera but instead to get a rough estimate of whether the person is close or far away from the camera to determine if the surrounding area is safe for UAV landing. Therefore, we choose a more straightforward solution that integrates well with the rest of our system and requires little computational power during inference time. Specifically, we leverage the bounding boxes information from the object detector as input for a regression model to predict the people's distance to the camera. The regression model of choice is the gradient-boosted decision trees algorithm implemented with the XGBoost library (Chen and Guestrin, 2016). The input for the distance estimator model is the bounding box representation of each object of interest, i.e., the person, and the output is their distance to the camera (for more details about data preparation, we refer the reader to Section 4.1). We adopt the format $\{x, y, w, h\}$ to represent each bounding box, where $\{x, y\}$ is the center coordinate of each box, and $\{w, h\}$ are its dimensions.

While previously shown in Figure 6 that the bounding box areas are correlated to the distance, better results can be obtained when inferring with other bounding box details, including its center point coordinates and its dimensions, since two pictures showing the same person at the same distance to the camera can have different bounding box shapes when the orientation changes. Furthermore, varying poses, e.g., crouching and sitting, can drastically change the shapes of the bounding boxes as well.

3.1.3 Vision-based localization

The bounding boxes from the object detector can provide insight into the relative orientation of a detected person to the camera, and the distance predicted by the XGBoost model can estimate how far they are from the camera. Fusing these two pieces of information allows a person to be sufficiently localized in a two-dimensional space. Initially, the image coordinate system must be transformed to one that matches the camera's coordinate system. To simplify the experiment, we position the camera and vehicle to align their coordinate axes with the world coordinate system (in this case, the coordinates of the MOCAP system). In the standard image coordinate system, the origin lies in the top left corner, with a horizontal x-axis from left to right and a vertical y-axis pointing downwards. To transform the image coordinates (in pixels) to the camera coordinate system depicted in Figure 2, the transformations

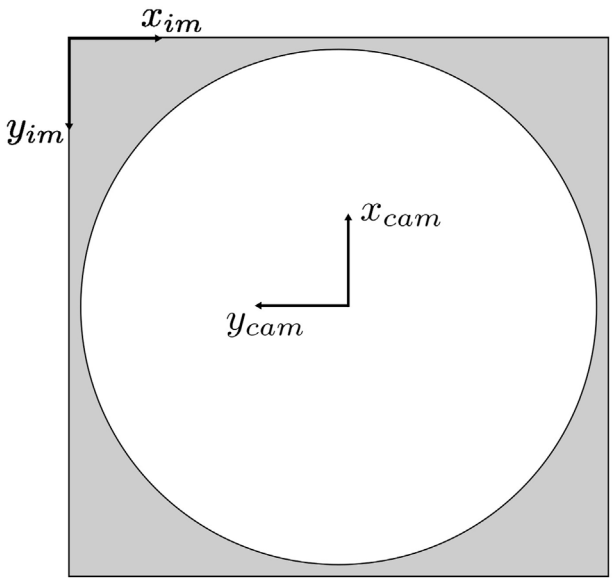

FIGURE 2
Image and camera coordinate systems.

are as follows:

$$x_{cam\_pix} = -y_{image} + \frac{w_{image}}{2}$$
$$y_{cam\_pix} = -x_{image} + \frac{h_{image}}{2} \quad (1)$$

Suppose the camera is not aligned with the world coordinate system. In that case, the offset angle between the world's and the camera's coordinate system must be pre-known, and a two-dimensional rotation transformation must be performed to obtain the coordinates of the detected people with respect to the world coordinate system on which the vehicle's local position and orientation are based.

From the transformed coordinate system, the orientation of the directional vector pointing to detected people can be obtained. For simplicity, we define this directional vector as the normalized vector from the camera coordinates origin to the center point of the corresponding bounding box. Then, the predicted distance from the XGBoost model is used to scale this vector to an approximate position of the detected person in two-dimensional space. We denote the coordinates of a person in the world coordinate system as X_p for simplicity, and the formula for calculating it is:

$$X_p = \begin{bmatrix} x_p \\ y_p \end{bmatrix} = d_{pred} \frac{\begin{bmatrix} x_{cam\_pix} \\ y_{cam\_pix} \end{bmatrix}}{\left\| \begin{bmatrix} x_{cam\_pix} \\ y_{cam\_pix} \end{bmatrix} \right\|} + \begin{bmatrix} d_{x\_cam} \\ d_{y\_cam} \end{bmatrix} \quad (2)$$

d_{x_cam} and d_{y_cam} are the camera's position in the world coordinate system, and d_{pred} is the distance prediction result from the distance estimator. Figure 3 summarizes the localization process, from acquiring the people's coordinates in a frame to projecting them into the world coordinates.

3.2 Safe landing program

3.2.1 System behavior

In our deployment, both in the Gazebo simulation and in a real-world indoor experiment, the vehicle operates in PX4's *offboard* mode, which allows full autonomy. We refer the reader to the PX4 paper or related documentation for more details about flight mode descriptions and their requirements (Meier et al., 2015). Our proposed system is designed with the assumption that the vehicle's flight stack, PX4, receives a steady stream of positional data to operate at all time. In cases of failure during our experiments, such as drifting or loss of positional data, we revert to *position* or *altitude* mode to attempt manual landing. In indoor environments, the UAV utilizes local coordinates for localization and determining mission setpoints. The experimental flight mission consists of four phases: taking off, performing the flight mission, pre-landing, and landing. The first, second, and last phases are self-explanatory, while the pre-landing phase activates the safe landing mechanism. We define pre-landing as going to a setpoint at a height safe for complete landing while continuously communicating over a ROS network with the detection module on the ground for information of the surrounding as shown in Figure 4.

During pre-landing, the vehicle will retreat to a safe position and hover if the detection module detects a person within a

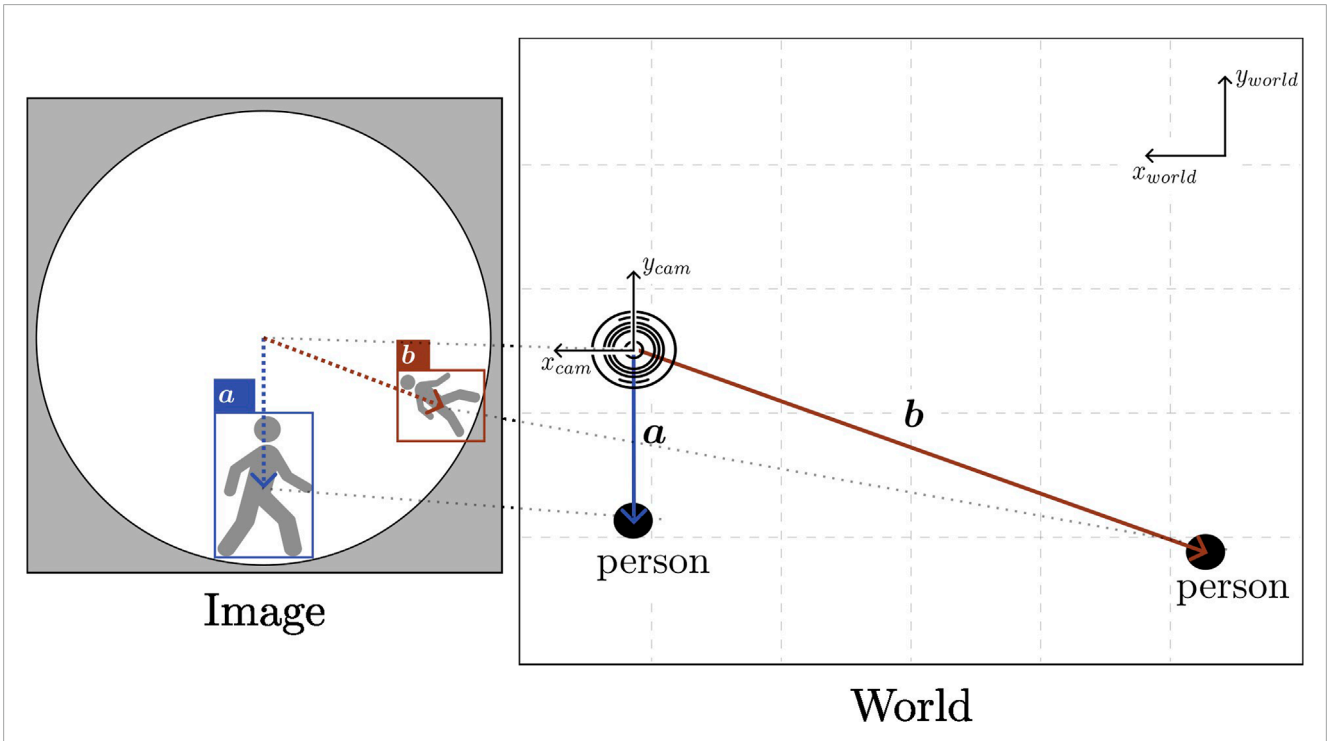

FIGURE 3
Illustration of how image coordinates are transformed to positions of detected people. Because of the panoramic nature of the sensor, images are heavily distorted near the ground level or image edges.

predefined safe threshold. After a set period, the vehicle switches to adaptive emergency landing mode and searches for an optimal landing position. The vehicle can also resort to this behavior in circumstances where hovering is impossible, e.g., when the payload is over a threshold or when the battery is low. We assume that aside from people around the landing area, there are no other direct threats to the landing procedure.

3.2.2 Adaptive emergency landing

When the vehicle can no longer hover at a safe position and must land immediately, the optimal position for landing, $X_o = \{x_o, y_o\}$), considering the vehicle's current position, which is also the camera's position, must satisfy multiple criteria. Firstly, it must move as far away from the surrounding people as possible. Secondly, its landing position must be away from each person by a specific range. Last but not least, we must ensure that the landing position is within a certain threshold, so the search range needs to be limited. To satisfy all requirements mentioned above, we reformulate the optimal landing position search into an optimization problem and utilize a solver to get the results. We implemented the landing spot search with SciPy's minimize function with the optimization method Sequential Least Squares Programming (SLSQP), which is suitable for constrained optimizations. Figure 5 illustrates how we approach the problem.

The positions of detected people are denoted as $X_1, \ldots, X_{n_p}$, and the position of the camera, which is also the hovering position of the UAV, is denoted as X_c. We define a *search zone* as a circular area with radius r_l where the optimizer can search for a landing spot. Around each detected person is a *danger zone* with range r_d, which the vehicle should avoid. Finally, the *scan zone* with range r_s is where all the people are considered to be in danger and should be avoided. The *scan zone* is also the area in which the emergency state for the flight controller is triggered, causing it to retreat the vehicle to a safe position and hover initially. To ensure that the vehicle does not go out of the *scan zone*, where the camera and the detection module do not provide enough information to conclude whether there are people, we restrict that $r_l \leq r_s$.

Because we want the UAV to land as far as possible from the people standing in close vicinity of the camera, i.e., the UAV's initial landing spot, for n_p humans detected in the *scan zone*, the function that the optimization solver must maximize is as follows:

$$\underset{x_o, y_o \in [-r_l, r_l]}{\operatorname{argmax}} \sum_{i=1}^{n_p} \|X_o - X_i\| \tag{3}$$

Then, to ensure the solution is not within the *danger zone*, the first constraint is formulated as:

$$\|X_o - X_i\| \geq r_d, \ \forall i \in \{0, \ldots, n_p\} \tag{4}$$

Lastly, the selected landing position must be in the predefined *search zone*:

$$\|X_o - X_c\| \leq r_l \tag{5}$$

While maximizing the function 3 results in a landing position that is the furthest from all detected people, it is sometimes safer to emphasize the people who are closer to the camera, which is also the hovering position of the UAV. To do so, we introduce another term to address how close a person is to function 3, and rewrite it as:

$$\underset{x_o, y_o \in [-r_l, r_l]}{\operatorname{argmax}} \sum_{i=1}^{n_p} \frac{1}{\|X_i - X_c\|^{\alpha}} \|X_o - X_i\| \tag{6}$$

```
Function bbox_callback(msg):
    for box ∈ msg.bounding_boxes do
        if box.dist ≤ r_s then
            EMERGENCY ← True
            danger_counter + +
            break
    /* If the vehicle has hovered for too long, switch to adaptive emergency
       landing                                                              */
    if danger_counter ≥ danger_threshold then
        if not EMERGENCY_LANDING then
            X_ppl ← [ ]                          /* List of people in the scan zone */
            for box ∈ msg.bounding_boxes do
                if box.dist ≤ r_s then
                    X_p ← localize(box)                          /* Section 3.1.3 */
                    X_ppl.append(X_p)
            X_o ← optim_search(X_ppl, x_s, x_r, x_l, α)          /* Section 3.2.2 */
            prelanding_pos[: 2] + = X_o
            EMERGENCY_LANDING ← True
while True do
    switch FLIGHT_MODE do
        case TAKEOFF do
            setpoint ← takeoff_position
        case MISSION do
            setpoint ← get_mission_setpoint(mission_setpoints)
        case PRELANDING do
            if EMERGENCY and not EMERGENCY_LANDING then
                setpoint ← retreat_pos                   /* Hover and wait */
            else
                setpoint ← prelanding_pos        /* Default or emergency landing */
        case LANDING do
            setpoint ← None land_command()
    if setpoint != None then
        publish_setpoint(setpoint)
```

Algorithm 1. Safe landing algorithm.

The parameter α controls how much the distance of each detected person in the *scan zone* to the UAV's current position impacts the selection of the landing spot. In other words, the higher α is, the more the UAV tries to avoid people close to it.

3.3 Offboard navigation

Algorithm 1 summarizes the safe landing program on the UAV's companion computer. It amalgamates the visual-based localization algorithm (see Section 3.1.3), the adaptive emergency landing algorithm (see Section 3.2.2), and a simple finite state machine determining each mission phase's setpoint (the phases are explained in detail in Section 3.2.1).

Several parameters related to the flight mission must be predetermined, including the take-off height, the mission waypoints, the retreat position, and the pre-landing position above the landing pad. Furthermore, the parameters for adaptive emergency landing mentioned in Section 3.2.2 should be tuned for different situations, including varying reliability of the detection module at different ranges, the available flight space, and the size of the UAV.

The condition mentioned in line 1 of the algorithm is used for the simulated and indoor experiments, which only sets a timeout period for the UAV's hovering. This condition can be extended to adapt to more types of emergencies that require immediate landing.

4 Experimental results

4.1 Data preparation

The inputs for the detector described in Section 3.1 are image frames collected from a single PICAM360 panoramic camera module. The images are retained in their original circular panoramic

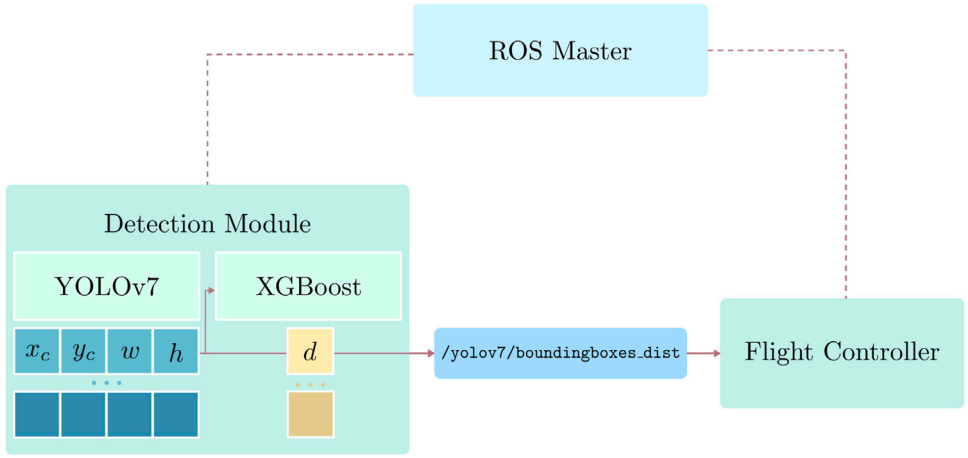

FIGURE 4
Communication between the detection module and the flight controller over a ROS network. Only the detection and position estimation results are sent to the UAV.

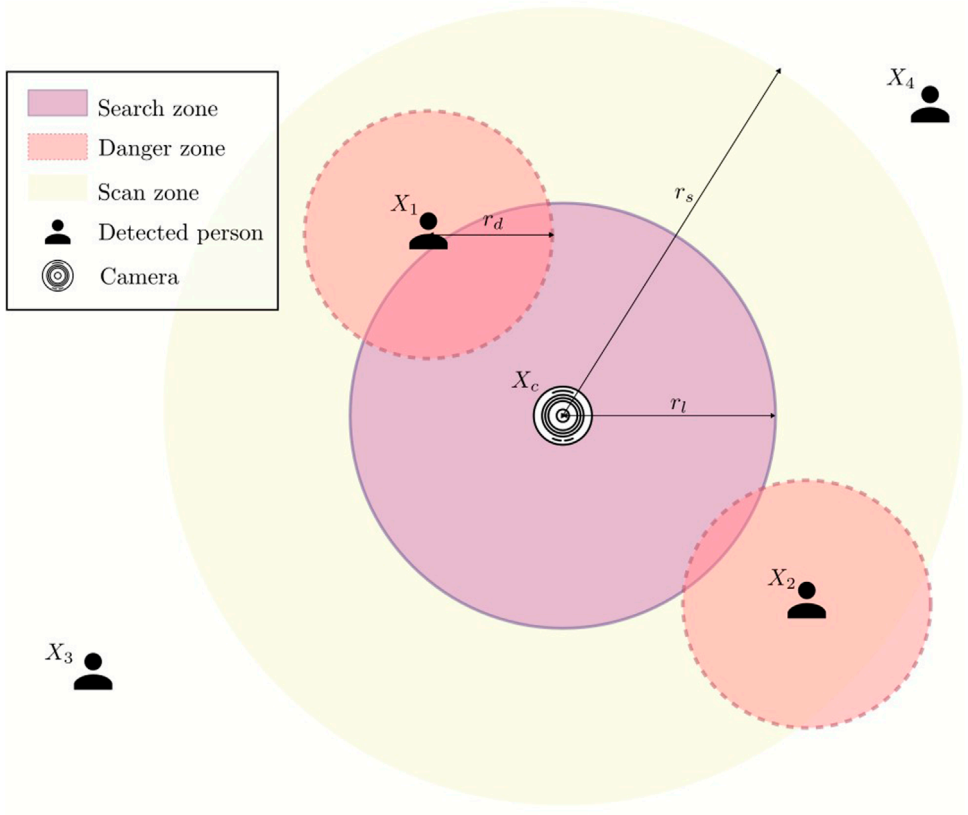

FIGURE 5
Optimization problem to solve to obtain safe landing position.

form to minimize the amount of pre-processing required and streamline implementation on embedded platforms. Our dataset comprises a training set with 5,062 images from 7 ROS bags and a test set with 2,030 images from 2 ROS bags.

To further enrich the dataset, data augmentation is a viable option that has proven effective in improving deep learning models' performance in various domains, including computer vision (Shorten and Khoshgoftaar, 2019; Kaur et al., 2021). We applied

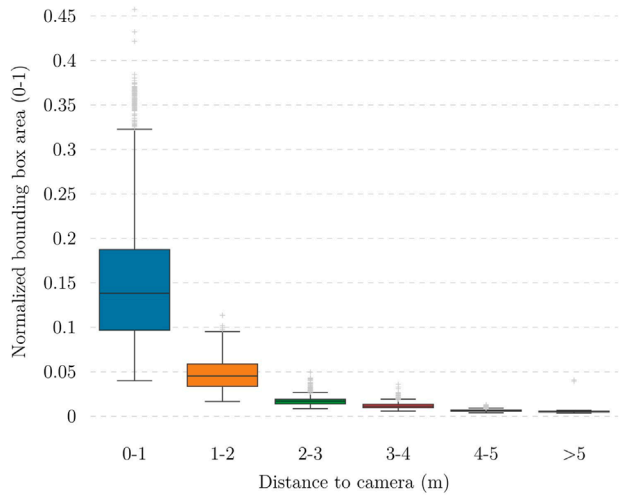

FIGURE 6
Relationship between normalized bounding box area and distance between the object to the camera. It is worth noting that high position accuracy is not needed; instead, a high recall in the detection (low probability of false negatives) and good classification accuracy (safe or unsafe distances) are more important.

rotational transformations to the original training and test sets with angle $\theta \in [90°, 180°, 270°]$. Conventionally, augmenting the test set is not advisable because it is crucial to maintain the authenticity of the unseen data that the model might encounter in the real world. However, rotating a circular fisheye image is valid in the context of this work as it simulates changes in the camera placement angle, which is very likely to happen in our application. Unlike pinhole cameras that have to be upfront when taking regular photos, the panoramic camera in this setup does not have to be in any specific orientation. Furthermore, while people are moving around the camera when the dataset is collected, the background does not, so hypothetically, this will also enhance the robustness of the trained models.

Another requirement for this project is estimating the distance between each detected human and the camera module. We experimented with two methods to get this information: Decawave's ultra-wideband (UWB) module DWM1001 and the Optitrack MOCAP system. Each person in the experiment holds a UWB module or a set of reflective markers; another module will be placed where the camera is. The distance between the person to the camera is calculated as the Euclidean distance in two-dimensional space between the module they are holding and the camera.

4.2 Human detector training details

4.2.1 Evaluation metric

The trained object detector should be able to reliably detect potential hazards to the landing operation, in this case, people around the landing site. Detecting people near the landing site is more critical as they pose a direct danger to the operation while simultaneously the UAV poses a hazard to nearby passers. Because the dimensions of the experimental flight zone are $9 \times 8m^2$, and the camera is placed near the middle of it during data collection and slightly shifted to the side during experiments, the furthest possible distance to the camera is approximately 5 m. We select 3 m as the safe range for the experiment, i.e., the distance that a detected person is considered close to the camera. To evaluate the trained model's performance on people at different ranges, we use a slightly modified version of COCO's AP across object size metric. Examining the distance data illustrated in Figure 6 shows that this distance negatively correlates with the normalized area of the corresponding bounding box, so it is reasonable to use the box area as a rough estimate to separate instances that are close to or far from the camera. The median bounding box area of samples within 3 ± 0.2 m vicinity of the camera from the dataset is approximately 0.0135, so the metrics that we use are:

- $\mathbf{AP^F}$: AP for far objects, bounding boxes with $area \geq 0.0135 h_{im} w_{im}$
- $\mathbf{AP^N}$: AP for near objects, bounding boxes with $area \leq 0.0135 h_{im} w_{im}$
- $\mathbf{AP^{all}}$: AP for all objects

4.2.2 Fine-tuning on panoramic dataset

To leverage the well-initialized weights of the pre-trained models, we use them as the foundation and fine-tune them on our training set. With this technique, it is possible to obtain a model capable of performing in the target environment with a relatively small amount of data compared to the large-scale COCO dataset. As mentioned in Section 3.1.1, we focus on training two model versions, YOLOv7-tiny and YOLOv7. Both models are trained with base inference size 640 and multi-resolution. Multi-resolution training is a technique that varies the training resolution to ±50{%} of the base resolution, which should improve the model's robustness to scaling changes and prediction performance on small objects. Previous research has reported promising results on this training technique's effectiveness in improving object detection models' resolution scalability (Yan et al., 2013; Tian et al., 2022).

4.2.3 Ablation studies

We now analyze the effect of data augmentation and multi-resolution training in the performance of the trained model.

4.2.3.1 Rotational augmentation

When forming the dataset, we apply rotational augmentation to enrich the dataset. We conducted this ablation study by evaluating the performance of the fine-tuned models on both the unaugmented and augmented datasets. Table 1 shows that the models trained on the augmented dataset outperform those trained only on unrotated data in all test cases.

4.2.3.2 Multi-resolution training

We utilized multi-resolution training to improve inference accuracy and robustness to scaling changes during the training process. As shown in Table 1 this training method improves the

TABLE 1 Performance of the fine-tuned (FT) models and ablation study on the effect of multi-resolution (MR) training and training on rotationally augmented data (Aug). As highlighted, both YOLOv7-tiny and YOLOv7 models perform better than their baseline counterparts when fine-tuned on our rotationally augmented dataset with multi-resolution training.

Model	AP^{all}	AP^{N}	AP^{N}_{50}	AP^{N}_{75}	AP^{F}	AP^{F}_{50}	AP^{F}_{75}
Unaugmented test set							
YOLOv7-tiny (**baseline**)	0.068	0.173	0.361	0.115	0.019	0.041	0.013
YOLOv7-tiny (**FT**)	0.316	0.529	0.937	0.531	0.238	0.490	0.201
YOLOv7-tiny (FT + Aug)	**0.388**	**0.528**	**0.955**	0.527	**0.327**	**0.603**	**0.337**
YOLOv7 (**baseline**)	0.133	0.321	0.661	0.263	0.045	0.117	0.032
YOLOv7 (**FT**)	0.393	0.535	0.941	0.545	0.327	0.664	0.301
YOLOv7 (FT + Aug)	**0.415**	**0.572**	**0.961**	**0.615**	**0.343**	**0.601**	**0.377**
Augmented test set							
YOLOv7-tiny (**baseline**)	0.098	0.174	0.392	0.118	0.066	0.160	0.044
YOLOv7-tiny (**FT**)	0.306	0.490	0.927	0.453	0.230	0.478	0.182
YOLOv7-tiny (**FT + Aug**)	0.380	0.513	0.944	0.519	0.318	0.576	0.340
YOLOv7-tiny (FT + MR + Aug)	**0.394**	**0.532**	**0.966**	**0.534**	**0.334**	**0.602**	**0.354**
YOLOv7 (**baseline**)	0.187	0.324	0.663	0.267	0.126	0.287	0.085
YOLOv7 (**FT**)	0.368	0.510	0.936	0.485	0.303	0.594	0.287
YOLOv7 (**FT + Aug**)	0.419	0.560	0.974	0.594	0.354	0.630	0.376
YOLOv7 (FT + MR + Aug)	**0.426**	**0.572**	**0.969**	**0.612**	**0.359**	**0.629**	**0.390**

TABLE 2 Frequency of /yolov7/boundindboxes_dist ROS topic at inference time with Pytorch and TensorRT implementations on an NVIDIA Jetson Xavier NX.

Object detection model	Pytorch		TensorRT	
	Visual data sent	No visual data sent	Visual data sent	No visual data sent
YOLOv7-tiny	15 Hz	20 Hz	28 Hz	30 Hz
YOLOv7	~	5 Hz	18 Hz	20 Hz

TABLE 3 Performance comparison between XGBoost models with default and tuned hyperparameters.

Hyperparameters	MAE	MedAE	MaxErr	ExpVar
Default	0.208	0.183	0.933	0.959
Tuned	0.199	0.159	0.825	0.961

model's performance compared to the model trained with fixed resolution.

4.2.4 Deployment on embedded platform

When the models are well-trained on the custom dataset and ready for deployment, they are converted to TensorRT engines and deployed on the target embedded platform. We select the NVIDIA Jetson Xavier NX development kit as the platform for deploying the detection module by virtue of its compact footprint and high AI performance. When integrating with the ROS detection node running on the ground computing platform, the ROS bounding box and distance messages must be published at a high frequency to address the tight latency requirements of real-time applications. The maximum frequency this message can be published is 30Hz, the highest supported framerate of the PICAM360 module. Because the XGBoost prediction has little to no effect on the topic's frequency during our experiments, the essential factor in the detection module's speed is the object detector's throughput. Table 2 shows the frequency of the bounding box and distance messages when the ROS node is running on the target

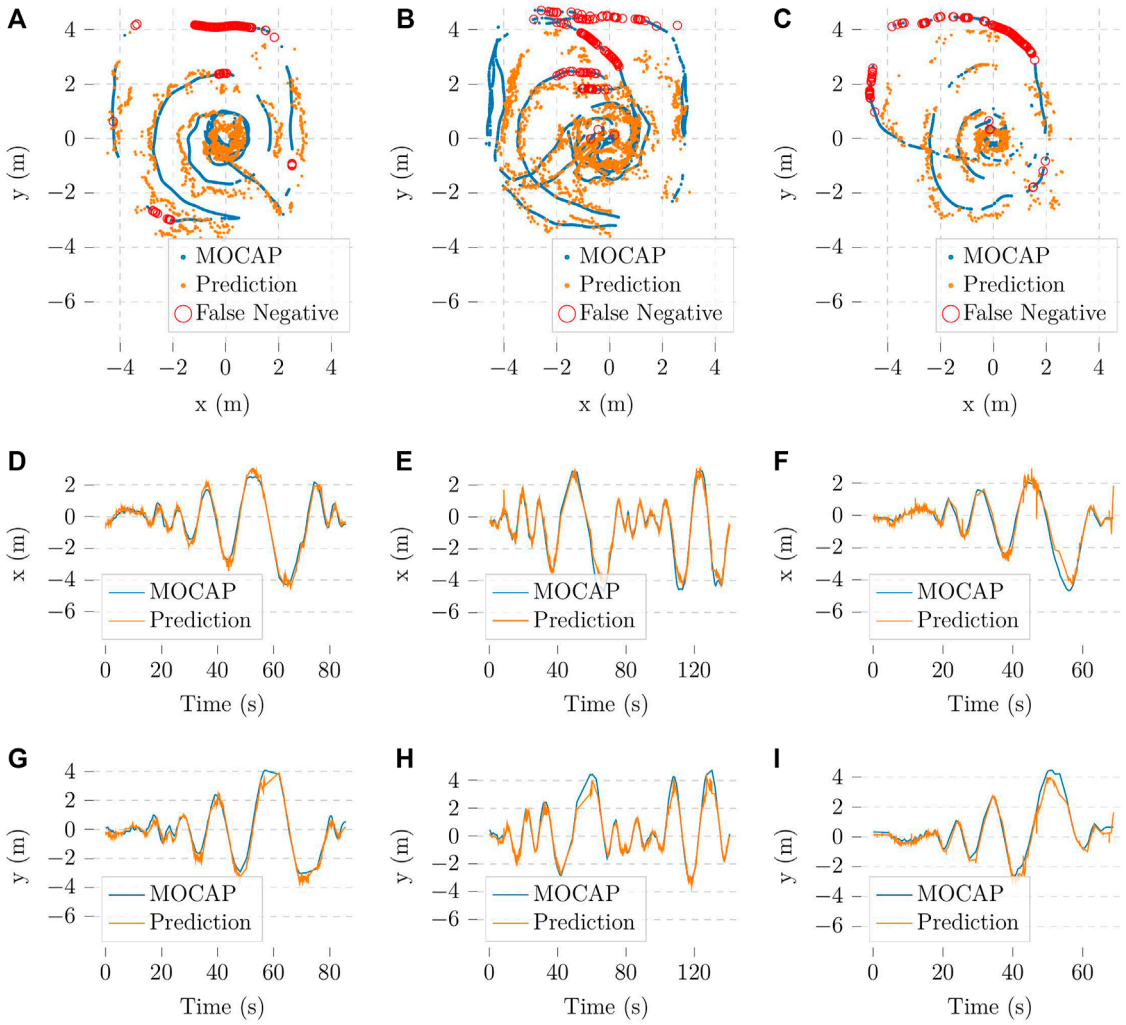

FIGURE 7
Vision-based localization trajectories in comparison with the trajectories from the Optitrack system **(A–C)** in the xy plane corresponding to the flight zone, **(D–F)** in the x-axis and **(G–I)** in the y-axis; missed detections are omitted in **(D–I)**.

embedded computer. We gather these measurements using the `rostopic` tool.

4.3 Distance estimator training details

We divided the training data into five sets of bounding box data from 5 different ROS bags. To optimize and validate the performance of the distance estimator, we keep one holdout set and perform a randomized search with cross-validation (Scikit-learn's `RandomizedSearchCV`) using the training set to obtain the optimal hyperparameters as follows:

- `max_depth`: 3
- `learning_rate`: 0.05
- `n_estimators`: 500
- `colsample_bytree`: 0.5
- `colsample_bylevel`: 0.8
- `subsample`: 0.6

These hyperparameters slightly improve the performance on the holdout set over the default ones, and the results in mean absolute error (MAE), median absolute error (MedAE), maximum error (MaxErr), and explained variation (ExpVar) are shown in Table 3.

4.4 Evaluation of vision-based localization algorithm

As both components of the detection module have been trained and tested, we proceed to evaluate the performance of the localization algorithm base on visual information. To simplify the evaluation process, we select three datasets with one person walking around the camera for testing. Three temporary XGBoost models were also trained without the test sets to avoid high accuracy

Trajectory	APE	Cossim
$\mathcal{R}1$	0.369	0.922
$\mathcal{R}2$	0.337	0.963
$\mathcal{R}3$	0.313	0.941

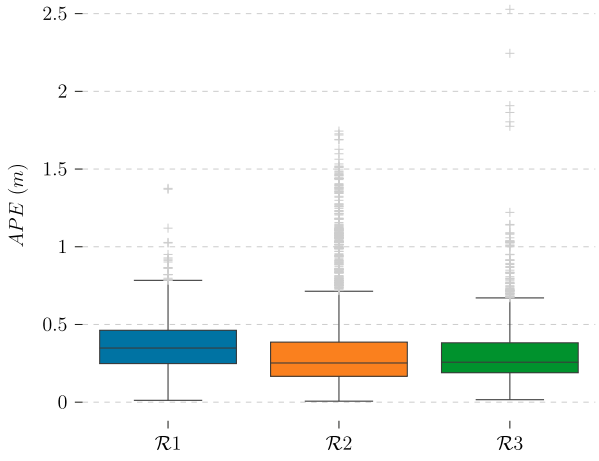

FIGURE 8
Evaluation of vision-based localization method. Each of the labels $\mathcal{R}_i$ represent three different experiments.

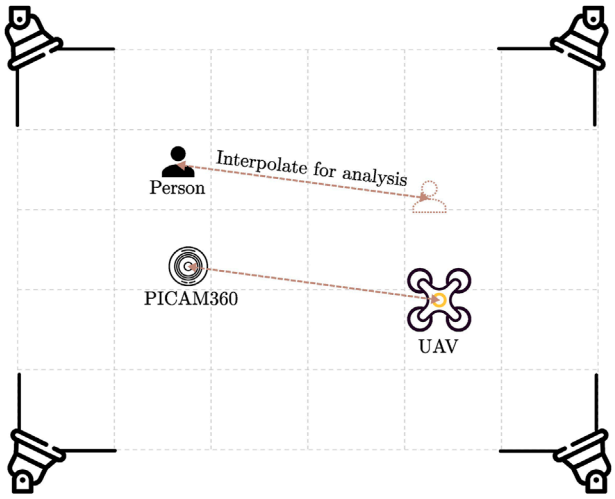

FIGURE 9
Experimental setup. For safety reasons during the experiments, the camera is not located directly where the UAV is landing. This allows for more experimental setups while staying at a safe distance from the UAV.

due to overfitting. The object detection model used for this test was the YOLOv7-tiny. After applying the algorithm mentioned in Section 3.1.3, the resulting trajectories are recorded and visualized in Figure 7.

We evaluate these results with cosine similarity (Cossim) and average positioning error (APE) metrics. The former gives insight into how accurate our method is at determining the direction in which the person is with respect to the camera. The latter quantifies how accurate the predicted trajectories in Figure 7 are. The experimental results are shown in Figure 8. For simplicity, the missing bounding boxes and the frames without corresponding Optitrack data are omitted when calculating the metrics.

4.5 Autonomous landing experiments

4.5.1 Simulation

The autonomous flight programs are thoroughly tested in a simulation environment before deployment to guarantee safety. The simulation environment was implemented with PX4 Gazebo SITL. The tests are conducted on a Laptop with an NVIDIA RTX3070 GPU to run the YOLOv7 models on image frames from the PICAM360. From the simulated results, we validate that both designed behaviors, hovering and adaptive emergency landing, function correctly during the pre-landing phase.

4.5.2 Experiments with real UAV

We conduct experiments within our flight zone to test the system's performance. The object detection model used in the detection module is YOLOv7-tiny. The parameters for the emergency landing algorithm are: $r_l = 1m, r_s = 3m, r_d = 0.5m$, and $\alpha = 0$ (see Section 3.2.2 for more information on the algorithm). As explained in Section 4.2.1, the safe distance from the camera, i.e., r_s, is 3 m based on the size of the experimental zone. We deliberately choose a small range for the *search zone*, i.e. r_l, to keep the experimental UAV within the flight zone. Furthermore, to simplify the experiment, the mission only consists of the UAV taking off and landing at the same spot afterward because we are most interested in the latter's behavior for the scope of this thesis. Because the safe landing software is still in development, to ensure the safety of the people involved in the experiments, as well as to protect the equipment of the on-ground monitoring system, we place the embedded computer running the detection module and the panoramic camera away from the UAV during the experiment and interpolate the positions of the camera and people to the UAV's position while analyzing the results. The experimental setup is described in Figure 9.

4.5.3 Emergency landing experiments

To assess the emergency landing in real-world experiments, we analyze the behaviour of the UAV in two different scenarios, with a single person and with two persons approaching the camera in multiple directions, respectively. It is worth noting that all computation runs in real-time during the experiments, including the landing location optimization. Figure 10 presents the optimal landing position selection results for clearer visualization. In Figure 10C, while the optimal landing position maximizes the distance to the detected people, it does not prioritize the closest person to the UAV. As explained in Section 3.2.2, this behavior can be altered by modifying the parameter α in Function 3 to increase the prioritization of the algorithm on the distance of the detected person to the camera/UAV's position. This effect is demonstrated in Figures 10C–F, which shows that increasing α increases the distance between the landing position and the closest person. In summary, the hovering and adaptive emergency landing behaviors work as expected in all experiments.

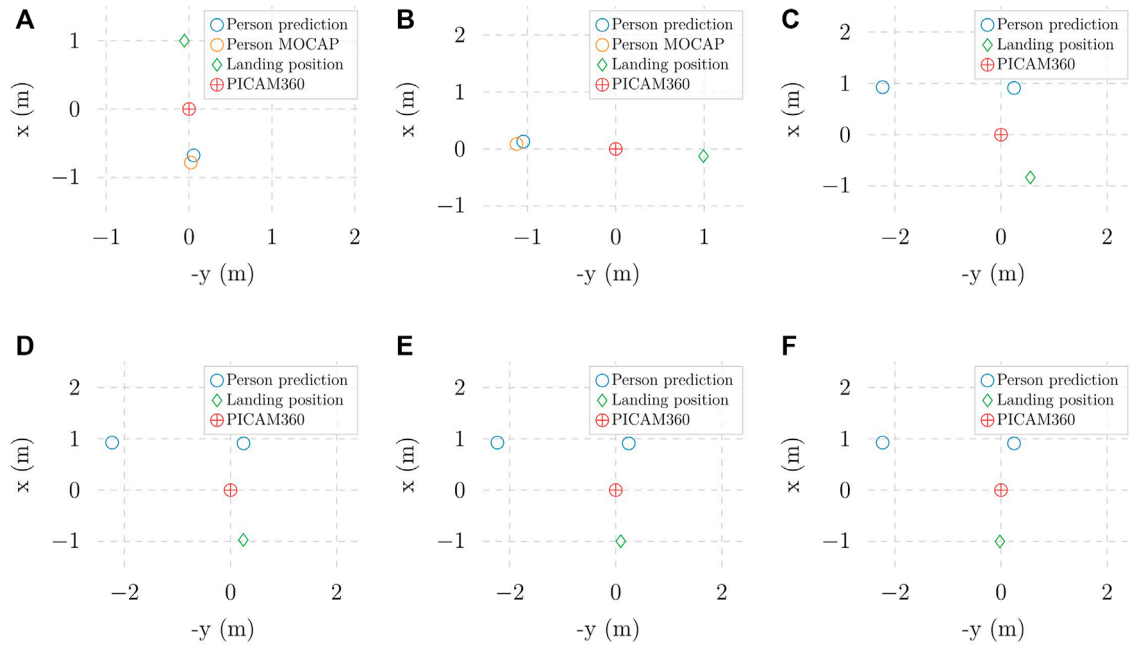

FIGURE 10
Optimal landing positions in different experiments with **(A, B)** one person approaching and **(C–F)** two people aproaching: **(C)** $\alpha = 0$ **(D)** $\alpha = 1$ **(E)** $\alpha = 1.5$ **(F)** $\alpha = 2$.

The experiments, both in simulation and in real-world indoor environments, are recorded in a video and uploaded at https://www.youtube.com/watch?v=XdolUS1bUVs.

5 Discussions and conclusion

5.1 Discussions

The current YOLOv7 models in this project were only trained on datasets consisting of people, which limits the system's functionality when dealing with other objects. In future work, we will expand the capability of the detection module by including more classes in the dataset. The object detection model in this work represents detected objects with standard horizontal bounding boxes. While the prediction performance from experimental results demonstrated that this representation is good enough for our implementation, the works of Rashed et al. (2021) and Xu et al. (2021) show that alternative shapes such as oriented boxes or ellipses can improve prediction accuracy and object coverage while having minimal impact on the model's inference speed. Furthermore, oriented representations can potentially benefit the distance estimation method used in this work because, unlike horizontal boxes, the areas of the boundaries are the same even if the detected object is oriented.

Another point that can be improved is the adaptive landing algorithm. We only perform optimization based on the information obtained before the UAV switches to adaptive emergency landing mode. While the solution is reliable when surrounding people are stationary, when they are moving, it would be better to incorporate tracking data into the formulation of the optimization problem. A method that can be used for this algorithm and would seamlessly fit into the current solution is tracking-by-detection (Bochinski et al., 2017). Tracking-by-detection is an object-tracking paradigm that leverages detection results from an object detection model like YOLO to track objects in video streams. Tracking data can enable the integration of model predictive control (Camacho and Alba, 2013), which obtains inference of future data points based on observed samples in the past to control a process. After that, the predicted positions of surrounding people can replace their current positions in Function 3.

5.2 Conclusion

In this paper, we propose a novel on-ground vision-based solution for safe UAV landing by leveraging the omnidirectional view capability of panoramic sensors. The detection module, comprising a YOLOv7-based object detector and an XGBoost-based distance estimator, demonstrates high capability in detecting and localizing humans near the landing zone while delivering real-time performance. Furthermore, a series of indoors experiments has proven the system's reliability in enabling landing UAVs to avoid surrounding pedestrians. Rather than completely replacing available onboard methods (Marcu et al., 2018; Tovanche-Picon et al., 2022), our solution serves as an extra layer of safety for UAV landing applications. Our ultimate goal is a collaborative autonomy approach where sensor and detection data from the micro-airports is fused with the UAVs' sensors and computational capabilities to enhance the system's reliability, safety, and efficiency.

Data availability statement

The raw data supporting the conclusion of this article will be made available by the authors, without undue reservation.

Ethics statement

Written informed consent was obtained from the individual(s) for the publication of any potentially identifiable images or data included in this article.

Author contributions

PN and JP conceived and conceptualized the work. PN elaborated the methods and implemented the software. PN was responsible of hardware and data collection. PN carried out the data processing and analysis. PN and JP conceived and illustrated the figures. JP and TW supervised the content. All authors contributed to the article and approved the submitted version.

Funding

This research work has been supported by the Academy of Finland's AeroPolis project (Grant No. 348480).

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

References

Alam, M. S., and Oluoch, J. (2021). A survey of safe landing zone detection techniques for autonomous unmanned aerial vehicles (UAVs). *Expert Syst. Appl.* 179, 115091. doi:10.1016/j.eswa.2021.115091

Bhowmik, D., and Appiah, K. (2018). "Embedded vision systems: a review of the literature," in Applied Reconfigurable Computing. Architectures, Tools, and Applications: 14th International Symposium, ARC 2018, Santorini, Greece, May 2-4, 2018, 204–216.

Bochinski, E., Eiselein, V., and Sikora, T. (2017). "High-speed tracking-by-detection without using image information," in 2017 14th IEEE international conference on advanced video and signal based surveillance (AVSS) (IEEE), Lecce, Italy, Aug. 29 2017 to Sept. 1 2017, 1.

Camacho, E. F., and Alba, C. B. (2013). *Model predictive control*. Berlin, Germany: Springer science & business media.

Castellano, G., Castiello, C., Mencar, C., and Vessio, G. (2020). "Crowd detection for drone safe landing through fully-convolutional neural networks," in SOFSEM 2020: Theory and Practice of Computer Science: 46th International Conference on Current Trends in Theory and Practice of Informatics, SOFSEM 2020, Limassol, Cyprus, January 20–24, 2020, 301–312.

Catalano, I., Sier, H., Yu, X., Queralta, J. P., and Westerlund, T. (2023a). *Uav tracking with solid-state lidars: dynamic multi-frequency scan integration. arXiv preprint arXiv:2304.12125.*

Catalano, I., Yu, X., and Queralta, J. P. (2023b). *Towards robust uav tracking in gnss-denied environments: a multi-lidar multi-uav dataset. arXiv preprint arXiv:2310.09165.*

Chen, T., and Guestrin, C. (2016). "Xgboost: a scalable tree boosting system," in Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, San Francisco, USA, Aug 13, 2016-Aug 17, 2016, 785–794.

Elfwing, S., Uchibe, E., and Doya, K. (2018). Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Netw.* 107, 3–11. doi:10.1016/j.neunet.2017.12.012

Geiger, A., Lenz, P., and Urtasun, R. (2012). "Are we ready for autonomous driving? the kitti vision benchmark suite," in 2012 IEEE conference on computer vision and pattern recognition (IEEE), Providence, Rhode Island, USA, 16-21 June 2012, 3354.

Ho, H. W., de Croon, G. C., and Chu, Q. (2017). Distance and velocity estimation using optical flow from a monocular camera. *Int. J. Micro Air Veh.* 9, 198–208. doi:10.1177/1756829317695566

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., et al. (2017). *MobileNets: efficient convolutional neural networks for mobile vision applications.* CoRR abs/1704.04861.

Kaur, P., Khehra, B. S., and Mavi, E. B. S. (2021). "Data augmentation for object detection: a review," in 2021 IEEE International Midwest Symposium on Circuits and Systems (MWSCAS), Michigan, USA, Aug 8-11, 2021, 537–543.

Kendoul, F. (2014). Four-dimensional guidance and control of movement using time-to-contact: application to automated docking and landing of unmanned rotorcraft systems. *Int. J. Robotics Res.* 33, 237–267. doi:10.1177/0278364913509496

Kong, W., Zhou, D., Zhang, D., and Zhang, J. (2014). "Vision-based autonomous landing system for unmanned aerial vehicle: a survey," in 2014 International Conference on Multisensor Fusion and Information Integration for Intelligent Systems (MFI) (IEEE), Beijing, China, September 28-29, 2014, 1. –8. doi:10.1109/MFI.2014.6997750

Liao, Y., Xie, J., and Geiger, A. (2022). *KITTI-360: a novel dataset and benchmarks for urban scene understanding in 2d and 3d.* arXiv.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. (2017). "Focal loss for dense object detection," in Proceedings of the IEEE international conference on computer vision, Venice, Italy, 22-29 October 2017, 2980–2988.

Marcu, A., Costea, D., Licaret, V., Pîrvu, M., Slusanschi, E., and Leordeanu, M. (2018). "Safeuav: learning to estimate depth and safe landing areas for uavs from synthetic data," in Proceedings of the European Conference on Computer Vision (ECCV) Workshops, Graz, Austria, May 13, 2006.

Martínez, C., Campoy, P., Mondragón, I., and Olivares-Méndez, M. A. (2009). "Trinocular ground system to control uavs," in 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems (Ieee), St. Louis, USA, October 11-15, 2009, 3361. –3367.

Meier, L., Honegger, D., and Pollefeys, M. (2015). "Px4: a node-based multithreaded open source robotics framework for deeply embedded platforms," in 2015 IEEE international conference on robotics and automation (ICRA) (IEEE), Seattle, Washington, USA, 26-30 May 2015, 6235. –6240.

Milano, M., Primatesta, S., and Guglieri, G. (2022). "Air risk maps for unmanned aircraft in urban environments," in 2022 International Conference on Unmanned Aircraft Systems (ICUAS) (IEEE), Dubrovnik, Croatia, June 21-24, 2022.

Nex, F., Armenakis, C., Cramer, M., Cucci, D. A., Gerke, M., Honkavaara, E., et al. (2022). Uav in the advent of the twenties: where we stand and what is next. *ISPRS J. photogrammetry remote Sens.* 184, 215–242. doi:10.1016/j.isprsjprs.2021.12.006

Qingqing, L., Taipalmaa, J., Queralta, J. P., Gia, T. N., Gabbouj, M., Tenhunen, H., et al. (2020). "Towards active vision with uavs in marine search and rescue: analyzing human detection at variable altitudes," in 2020 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR) (IEEE), Abu Dhabi, United Arab Emirates, 4-6 November 2020, 65–70.

Ranftl, R., Lasinger, K., Hafner, D., Schindler, K., and Koltun, V. (2020). Towards robust monocular depth estimation: mixing datasets for zero-shot cross-dataset transfer. *IEEE Trans. pattern analysis Mach. Intell.* 44, 1623–1637. doi:10.1109/tpami.2020.3019967

Rashed, H., Mohamed, E., Sistu, G., Kumar, V. R., Eising, C., El-Sallab, A., et al. (2021). "Generalized object detection on fisheye cameras for autonomous driving: dataset, representations and baseline," in Proceedings of the IEEE/CVF Winter Conference

on Applications of Computer Vision, Waikoloa, HI, USA, Jan. 3 2022 to Jan. 8 2022, 2272–2280.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). "You only look once: unified, real-time object detection," in Proceedings of the IEEE conference on computer vision and pattern recognition, Las Vegas, NV, USA, June 27 2016 to June 30 2016, 779–788.

Redmon, J., and Farhadi, A. (2018). *Yolov3: an incremental improvement*. *arXiv preprint arXiv:1804.02767*.

Safadinho, D., Ramos, J., Ribeiro, R., Filipe, V., Barroso, J., and Pereira, A. (2020). UAV landing using computer vision techniques for human detection. *Sensors* 20, 613. doi:10.3390/s20030613

Shorten, C., and Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *J. big data* 6, 60–48. doi:10.1186/s40537-019-0197-0

Sier, H., Yu, X., Catalano, I., Queralta, J. P., Zou, Z., and Westerlund, T. (2023). "Uav tracking with lidar as a camera sensor in gnss-denied environments," in 2023 International Conference on Localization and GNSS (ICL-GNSS) (IEEE), Castellon, Spain, June 6-8, 2023.

Song, B. D., Park, K., and Kim, J. (2018). Persistent UAV delivery logistics: MILP formulation and efficient heuristic. *Comput. Industrial Eng.* 120, 418–428. doi:10.1016/j.cie.2018.05.013

Tian, R., Wu, Z., Dai, Q., Hu, H., Qiao, Y., and Jiang, Y.-G. (2022). *ResFormer: scaling ViTs with multi-resolution training. arXiv preprint arXiv:2212.00776*.

Tovanche-Picon, H., Gonzalez-Trejo, J., Flores-Abad, A., and Mercado-Ravell, D. (2022). *Visual-based safe landing for UAVs in populated areas: real-time validation in virtual environments*. *arXiv preprint arXiv:2203.13792*.

Vajgl, M., Hurtik, P., and Nejezchleba, T. (2022). Dist-YOLO: fast object detection with distance estimation. *Appl. Sci.* 12, 1354. doi:10.3390/app12031354

Wang, C.-Y., Bochkovskiy, A., and Liao, H.-Y. M. (2022). *YOLOv7: trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. arXiv preprint arXiv:2207.02696*.

Wang, W., Song, G., Nonami, K., Hirata, M., and Miyazawa, O. (2006). "Autonomous control for micro-flying robot and small wireless helicopter xrb," in 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems (IEEE), Beijing, China, 9-13 October 2006, 2906.

Xiang, H., and Tian, L. (2011). Development of a low-cost agricultural remote sensing system based on an autonomous unmanned aerial vehicle (UAV). *Biosyst. Eng.* 108, 174–190. doi:10.1016/j.biosystemseng.2010.11.010

Xu, Y., Fu, M., Wang, Q., Wang, Y., Chen, K., Xia, G.-S., et al. (2021). Gliding vertex on the horizontal bounding box for multi-oriented object detection. *IEEE Trans. Pattern Analysis Mach. Intell.* 43, 1452–1459. doi:10.1109/TPAMI.2020.2974745

Yan, J., Zhang, X., Lei, Z., Liao, S., and Li, S. Z. (2013). "Robust multi-resolution pedestrian detection in traffic scenes," in Proceedings of the IEEE conference on computer vision and pattern recognition, Portland, OR, USA, June 23 2013 to June 28 2013, 3033–3040.

Yang, T., Li, G., Li, J., Zhang, Y., Zhang, X., Zhang, Z., et al. (2016). A ground-based near infrared camera array system for uav auto-landing in gps-denied environment. *Sensors* 16, 1393. doi:10.3390/s16091393

Yogamani, S., Hughes, C., Horgan, J., Sistu, G., Varley, P., O'Dea, D., et al. (2019). "Woodscape: a multi-task, multi-camera fisheye dataset for autonomous driving," in Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, BC, Canada, Oct. 11 2021 to Oct. 17 2021, 9308.

Zhang, H., and Zhao, J. (2017). An integrated unmanned aerial vehicle system for vision based control. *Dyn. Syst. Control Conf.* 58295, V003T39A011. doi:10.1115/DSCC2017-5405

Zhao, C., Sun, Q., Zhang, C., Tang, Y., and Qian, F. (2020). Monocular depth estimation based on deep learning: an overview. *Sci. China Technol. Sci.* 63, 1612–1627. doi:10.1007/s11431-020-1582-8

Zhu, J., Zhu, J., Wan, X., Wu, C., and Xu, C. (2019). Object detection and localization in 3D environment by fusing raw fisheye image and attitude data. *J. Vis. Commun. Image Represent.* 59, 128–139. doi:10.1016/j.jvcir.2019.01.005

Zou, Z., Shi, Z., Guo, Y., and Ye, J. (2019). *Object detection in 20 years: a survey*. *arXiv preprint arXiv:1905.05055*.