DOI: 10.13009/EUCASS2025-349
11TH EUROPEAN CONFERENCE FOR AERONAUTICS AND SPACE SCIENCES (EUCASS)

Vision Based Object Detection for UAV Applications

Luca Scarioni[⋆]*, Alessandro Nazzari*[⋆†]*, Roberto Rubinacci*[⋆] *and Davide Invernizzi*[⋆]
[⋆]*Dipartimento di Scienze e Tecnologie Aerospaziali, Politecnico di Milano*
20156 Milan, Italy
{luca.scarioni, alessandro.nazzari, roberto.rubinacci, davide.invernizzi}@polimi.it
[†]Corresponding author

Abstract

Recent advancements have transformed Unmanned Aerial Vehicles (UAVs) from basic remote-controlled devices into autonomous systems with advanced sensors and communication, expanding their use from traditional monitoring and surveillance applications to innovative aerial services, with operations in dynamic and potentially unstructured environments. In this context, UAVs must accurately detect and track objects to enable obstacle avoidance and target tracking. This work investigates vision-based object detection and tracking algorithms tailored for UAV applications. We focus on the selection, integration, and experimental evaluation of several object detection algorithms to evaluate their performance and effectiveness.

1. Introduction

The use of UAVs has grown steadily over the past decade across various domains, including agriculture,[3] search & rescue,[15] delivery services,[5] disaster response,[13] and surveillance.[2] Despite this growth, achieving reliable autonomous operation remains a significant challenge. Object detection and tracking of obstacles are some of the most challenging tasks. The goal of the former is to estimate 3D positions of objects, *e.g.*, by representing them as 3D bounding boxes. The goal of the latter is to assign an estimated velocity and acceleration to an object at each time step. For the object detection task, we compare two approaches, those based on U-disparity maps and YOLO-based detection. To enable accurate tracking of multiple objects, we combine Kalman filtering with data association techniques. Specifically, to ensure correct object-to-measurement associations, especially in scenarios with multiple objects of the same class, the Hungarian algorithm[7] is considered, leveraging object labels from YOLO to aid in the association process. We perform an experimental campaign to test algorithms considering as targets to be detected a moving person and a small quadrotor. The experiments aim to evaluate the precision of the detection and the trajectory reconstructed by the algorithm of a moving object, comparing the results of our algorithm with the ground truth provided by a Motion Capture system (Mo-Cap).

1.1 State of the art

Object detection is a widely studied and continuously evolving research area. Its goal is to identify and localise objects within an environment using sensor data such as RGB images, stereo cameras, LiDAR point clouds, or radar outputs. Unlike pure classification tasks, object detection focuses on estimating the spatial location and physical dimensions of each object, a crucial requirement for applications like autonomous driving, robotics, and UAV navigation.

One class of object detection techniques involves clustering of 3D point cloud data. These methods exploit the geometric properties of point clouds to identify objects based on spatial distribution. Euclidean Clustering is one of the simplest and most intuitive approaches: points are grouped if they are within a fixed threshold distance from each other. Tordesillas et al.[17] apply this method in the PANTHER framework, enabling real-time onboard obstacle detection and trajectory prediction for drones. DBSCAN (Density-Based Spatial Clustering of Applications with Noise) offers an improvement by grouping points based on density rather than fixed distance, allowing for better handling of noise and irregular shapes. Eppenberger et al.[4] combine DBSCAN with a 2D object detector to enhance object identification in stereo camera point clouds. Although more computationally intensive, DBSCAN has proven more accurate than Euclidean Clustering in complex environments.

The rise of deep learning has significantly advanced object detection capabilities in 2D images. Early methods such as the Viola-Jones detector[18] and the Deformable Parts Model (DPM)[6] laid the groundwork for modern

DOI: 10.13009/EUCASS2025-349

approaches, albeit with limitations in processing speed and adaptability. Later, the advent of Convolutional Neural Networks (CNNs) marked a turning point, enabling the direct learning of features from raw data. Modern architectures such as You Only Look Once (YOLO),[11] SSD,[9] and Faster R-CNN[12] represent state-of-the-art performance in 2D detection. YOLO, in particular, has become popular for its real-time performance, detecting objects and predicting bounding boxes in a single forward pass, reducing the computational time required to detect objects. For this reason, it is well-suited for not only image but also video processing. However, these methods operate on 2D data and thus require additional mechanisms to estimate object depth and orientation in a 3D space. To address this, hybrid approaches have been proposed. Wang and Jia[19] introduce a technique that combines 2D detection with 3D point cloud data through the concept of a sliding frustum, using PointNet[1] for final classification.

Another class of techniques leverages stereo vision and disparity information to estimate depth. The U-Disparity method is a notable example, it originates from the automotive domain[8] and was later adapted for UAVs by Oleynikova et al.[10] It analyses the disparity map column-wise to detect objects based on peaks in disparity histograms, enabling fast and lightweight onboard processing. This method is further enhanced using V-Disparity to recover vertical positioning.[14] These approaches are especially suited for UAVs due to their low computational cost. However, they can suffer from limitations in detecting complex or irregularly shaped objects due to their reliance on uniform disparity assumptions.

2. Methodology

In this section, we present a detailed overview of the object detection and tracking framework used in this work. Specifically, we explain how 2D bounding boxes are obtained in the camera frame, how stereo camera data are fused to generate 3D bounding boxes, and how a Kalman filter is employed to track moving obstacles.

2.1 2D Object detection by YOLO

YOLO is a convolutional neural network designed for real-time object detection and classification in 2D images. It identifies objects within RGB images, as illustrated in Figure 1, by outputting a 2D bounding box along with a single confidence score that jointly reflects the certainty in both the object's class and its spatial location within the image.

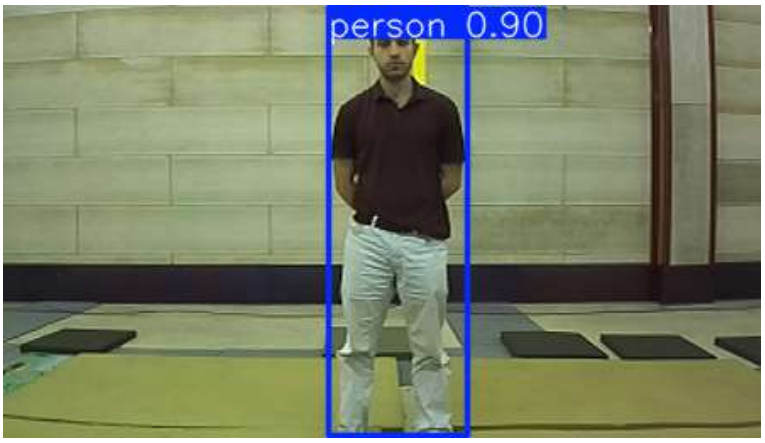


Figure 1: YOLO's result on the RGB images. The number indicates the level of confidence in the detection.

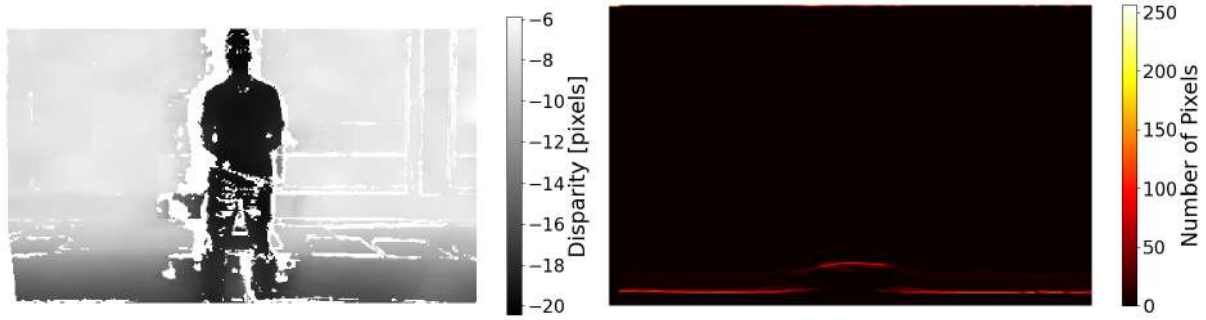
We employed the pretrained YOLOv8 model[16] developed by Ultralytics, which is already capable of detecting persons. To extend its capability to detect drones, the model was fine-tuned using 450 manually annotated images of multiple quadrotors flying at various distances. Through data augmentation techniques, including variations in lighting, hue, and rotation, the dataset was expanded to a total of $1,000$ images. These were split into 900 images for training, 80 for validation, and 20 for testing. The trained model demonstrates good performance. On the validation set, it fails only once to detect a drone, and shows no false positives.

2.2 3D Object detection by U-disparity map

Given two RGB images captured simultaneously by a stereo camera, it is possible to recover the 3D position of an object or point detected in both images. One common method involves computing a disparity map, where the disparity

DOI: 10.13009/EUCASS2025-349

represents the difference in pixel positions of the same point across the two images. An example of a disparity image is given in Figure 2, where closer objects exhibit higher disparity values, while farther objects have lower values. By counting the disparity values for each column of the disparity image, one obtains the U-Disparity map shown in Figure 2, which provides information about the distance and thickness of objects that stand out from the background.[10] For instance, Oleynikova[10] leveraged the U-disparity map to detect trees by fitting ellipsoids to clusters of disparity data, efficiently modelling the trees' 3D structure.



(a) Disparity image (b) U-Disparity map

Figure 2: Disparity data of a person standing still in front of the camera.

2.3 3D Object detection by YOLO

2.3.1 YOLO & U-Disparity

The main drawback of the U-Disparity method is the effect of noisy measurements on the detections. Background objects are often mistakenly included, and their disparity values can cause the generated bounding boxes to be larger than necessary. To address this issue, we propose combining the U-Disparity approach with YOLO by restricting the disparity analysis to the regions of the image where YOLO detects objects. This leads to faster and more accurate distance estimation. By integrating YOLO with the U-disparity approach, we combine the strengths of both methods: YOLO delivers accurate, class-labelled object detections, while U-disparity adds corresponding depth information. This fusion results in precise, categorised obstacle detections with associated distance estimates, making it well-suited for real-time applications.

2.3.2 Yolo MAD

An alternative approach consists of evaluating the median of the depth values within the 2D bounding box generated by YOLO detections and using the Median Absolute Deviation (MAD) to decide which pixels are part of the detected object. Specifically, let the MAD be defined as

$$\text{MAD} = \text{median}(|d_i - \tilde{d}|), \tag{1}$$

where d_i is the depth value associated with the i^{th} pixel in the region of interest and $\tilde{d}$ is the median of the values of depth present in the region of interest. Under the hypothesis that the median depth corresponds to the object itself (*i.e.*, more than half of the pixels in the bounding box belong to the object), it is possible to estimate a range of values around the median that indicate the object's thickness. Let

$$S_{MAD} = \{d_i : \tilde{d} - n \cdot MAD \leq d_i \leq \tilde{d} + n \cdot MAD\}, \tag{2}$$

where n is a tuning parameter, then the pixel whose depth belongs to S_{MAD} belongs to the object; this approach is denoted as YOLO-MAD and was introduced in.[20]

2.4 Tracking

To enable robust and accurate tracking of multiple objects, we combine Kalman filtering with data association techniques. To ensure correct track-to-measurement associations, especially in scenarios with multiple objects of the same class, the Hungarian algorithm has been utilised, leveraging object labels from YOLO to aid in the association process.

DOI: 10.13009/EUCASS2025-349

Assuming a constant velocity model, the Kalman Filter estimates the state (position and velocity) of each tracked object using the noisy measurements provided by the object detection module. Therefore, the process dynamics used in the Kalman Filter is a double integrator one, while we assume position measurements are available.

$$x_k = A \cdot x_{k-1} + w_{k-1}, \tag{3}$$

$$z_k = H \cdot x_k + v_k, \tag{4}$$

where x_k is the state vector at time k, representing the position and velocity of an object. A is the *state transition matrix*, seen in eq. 5, that relates the state at the previous time step to the current state. Δt represents the time elapsed between the two states. w_{k-1} represents the *process noise*, which captures random effects in the system dynamics, assumed to be normally distributed with covariance Q. z_k is the *measurement vector*, representing observed data at time k, while H is the *observation matrix* shown in eq. 6. The *measurement noise* v_k is assumed to be normally distributed with covariance R.

$$A = \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{6}$$

The Kalman Filter iteratively performs two steps, **prediction** and **update**, to refine the system state estimate using prior knowledge and new measurements.

Prediction Step:

$$\hat{x}_{k|k-1} = A \cdot \hat{x}_{k-1|k-1}, \quad P_{k|k-1} = A \cdot P_{k-1|k-1} \cdot A^T + Q. \tag{7}$$

The state $\hat{x}_{k|k-1}$ and covariance $P_{k|k-1}$ are predicted using the system model and process noise Q.

Update Step:

$$\begin{aligned} K_k &= P_{k|k-1} \cdot H^T \cdot (H \cdot P_{k|k-1} \cdot H^T + R)^{-1}, \\ \hat{x}_{k|k} &= \hat{x}_{k|k-1} + K_k \cdot (z_k - H \cdot \hat{x}_{k|k-1}), \\ P_{k|k} &= (I - K_k \cdot H) \cdot P_{k|k-1}, \end{aligned} \tag{8}$$

the Kalman Gain K_k adjusts the prediction using the innovation $z_k - H \cdot \hat{x}_{k|k-1}$, and the covariance $P_{k|k}$ reflects the confidence in the refined estimate.

In scenarios with multiple object, we use the hungarian algorithm to associate the new measurment to the correct trajectory. Pairing is only performed between tracks that share the same object label. Given a set of tasks (trajectories) and a set of workers (detections), each associated with a specific assignment cost (distance between predicted and detected positions), the algorithm identifies the optimal task-worker pairing that minimises the total assignment cost. Given a cost matrix C of minimum dimension n, where $C[i, j]$ represents the cost of assigning task (trajectory) j to worker (detection) i, the goal is to find a permutation π of $\{1, 2, \ldots, n\}$ such that:

$$\text{Minimize} \sum_{i=1}^{n} C[i, \pi(i)].$$

At each iteration, the algorithm constructs a solution based on the minimum values of each row and column, discarding associations that lead to a non-optimal solution and searching for new combinations that result in a lower overall cost. If the cost of assigning a detection to a track exceeds a predefined threshold, even if it represents the optimal pairing, the association is rejected. In such cases, a new track is initialised for the unmatched detection, while the original track remains temporarily active. Tracks that are not updated for a certain duration, as those that are not assigned any new detections, are eventually deactivated. Once a track is deactivated, the algorithm ceases to consider it for future associations. This strategy is particularly effective for handling objects that exit the drone's field of view. Furthermore, this mechanism is robust to the presence of outliers or erroneous detections. When such anomalies occur, the system may generate a new track; however, in the absence of subsequent supporting detections, these tracks are quickly discarded, thereby maintaining the integrity and reliability of the overall tracking framework.

DOI: 10.13009/EUCASS2025-349

3. Real world experiments

3.1 Experimental setup

The experiments took place at the Aerospace System & Control Laboratory (ASCL) of Politecnico di Milano, an indoor facility equipped with a Mo-Cap system. This system includes 12 infrared cameras mounted above the flight area, which track reflectors on each UAV and provide ground truth data for position and velocity. The drone used in the experiments, shown in Figure 3(a), is a quadrotor equipped with an NVIDIA Jetson companion computer and a ZED 2i stereo camera. It runs a Linux operating system configured with ROS2 (Robot Operating System), which enables modular and flexible deployment of robotic applications, including communication between sensors, actuators, and processing nodes. The algorithms have been implemented as ROS2 nodes in Python. The drone to be detected in the experiments is an ANT-X drone[1], a small quadrotor shown in Figure 3(b).

(a) Neurav drone

(b) Standard ANT-X quadrotor

Figure 3: The drones used in the experiment

3.2 First Experiment: detection of a single person

The first experiment aims to compare several object detection algorithms. The evaluated methods are the U-Disparity method, YOLO-MAD ($n = 15$), and YOLO & U-Disparity. Additionally, we consider a clustering-based approach, which generates axis-aligned bounding boxes by applying clustering to the point cloud data. We evaluate the following performance metrics:

- Intersection over Union (IoU). IoU measures the overlap between estimated and ground truth cuboids in 3D space, and is used to evaluate the precision of the detection method;
- Computational time.

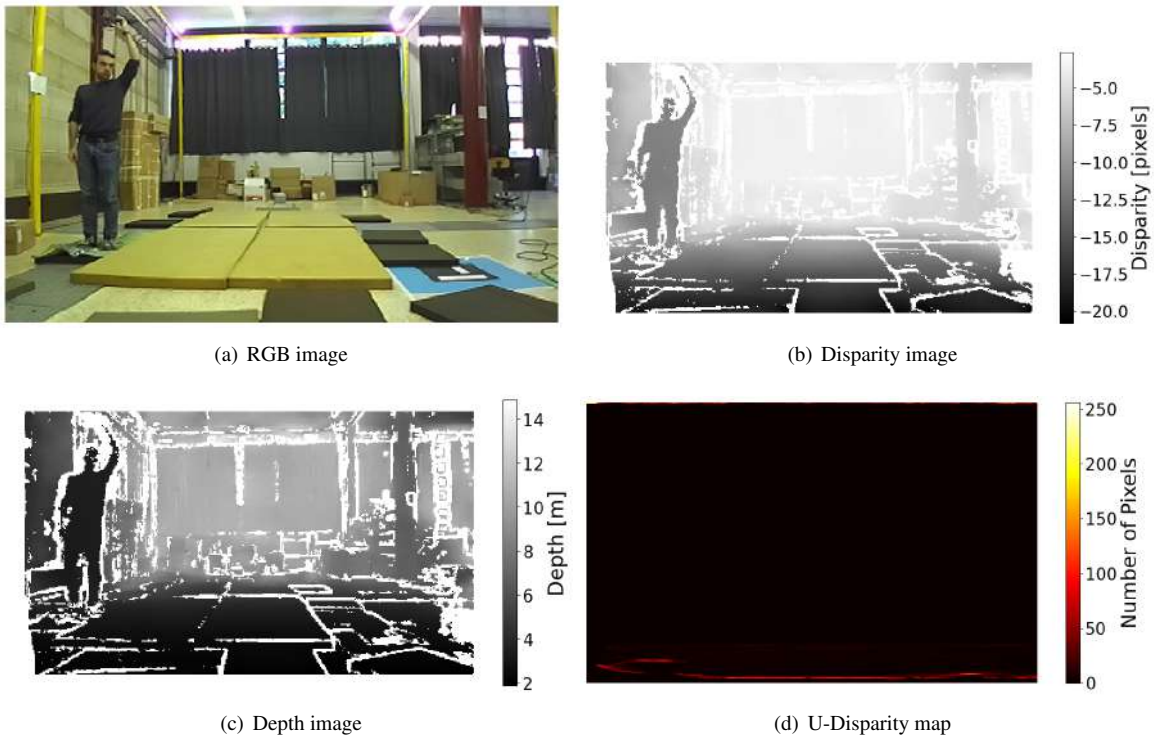
Table 1: Exp #1: Results

	Mean IoU	**CI IoU (95 %)**	**Mean Time [ms]**
Cluster	0.421	0.018	29.2
U-Disp.	0.249	0.012	5.0
YOLO & U-Disp.	0.390	0.016	39.0
YOLO-MAD	0.498	0.005	43.4

The results of the experiments are shown in Table 1. The table reports the mean and the half-width of the 95% confidence interval of the IoU, and the mean computational time. The experiment demonstrates that the YOLO-based methods, though requiring more computational time, provide better results than both the Cluster and U-disparity methods. Nonetheless, Table 1 shows that YOLO-based methods can run in real-time and are suitable for onboard use on a drone.

[1] `https://antx.it/` last accessed June 2025

DOI: 10.13009/EUCASS2025-349



(a) RGB image

(b) Disparity image

(c) Depth image

(d) U-Disparity map

Figure 4: Exp #2: The data necessary to the object detection methods, related to the same frame.

3.3 Second Experiment: Tracking of a single person

The second experiment focuses on the tracking of a person: the goal is to reconstruct the velocity and predict the position of the target. The person performs a square loop in front of the drone. In this experiment, YOLO-MAD and YOLO & U-Disparity have been tested.

Figure 5 shows the results of the experiment. Figures 5(a), 5(c) show the position of the target along the x and y component, respectively, along with the estimated position provided by the Kalman filter based on the observation of YOLO-MAD and YOLO & U-Disparity, while Figures 5(b), 5(d) show the velocity of the target and the estimated one along the x and y component, respectively.

The Root Mean Square Error (RMSE) between the predicted trajectory by YOLO-MAD and the ground truth is 0.21 m , while for YOLO & U-Disparity is 0.26 m.

3.4 Third Experiment: tracking of two people

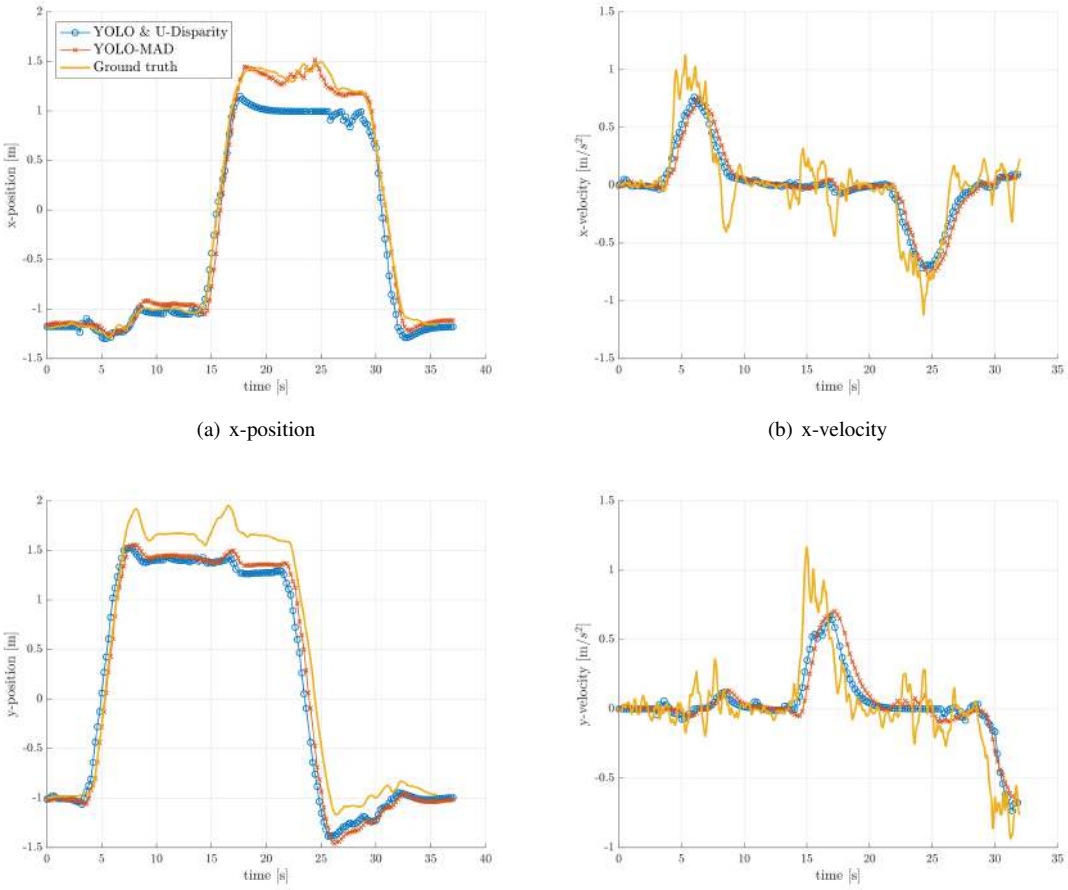
The third experiment aims at tracking multiple objects. It consists of two people moving toward the drone. While YOLO is capable of simultaneously detecting two people, it produces the same label for both. We use the Hungarian algorithm to assign the correct measurement to each trajectory at each time step. Due to the better performance shown in the previous experiment, we use YOLO-MAD to obtain 3D relative position measurements.

Figure 6 shows the trajectories, obtained using the Mo-Cap and the estimated ones, Figure 6(a) shows the x position while Figure 6(b) shows the y position. The RMSE for the former is 0.2183 m while the latter recorded a RMSE of 0.1917 m.

3.5 Fourth Experiment: tracking of a drone

This experiment focuses on tracking a small flying quadrotor. The same procedure used in Experiment 1 is applied to a drone, which performed a square trajectory in front of the camera. YOLO-MAD provides a good result, as seen in Figure 7: it achieves 0.37 meters of RMSE.

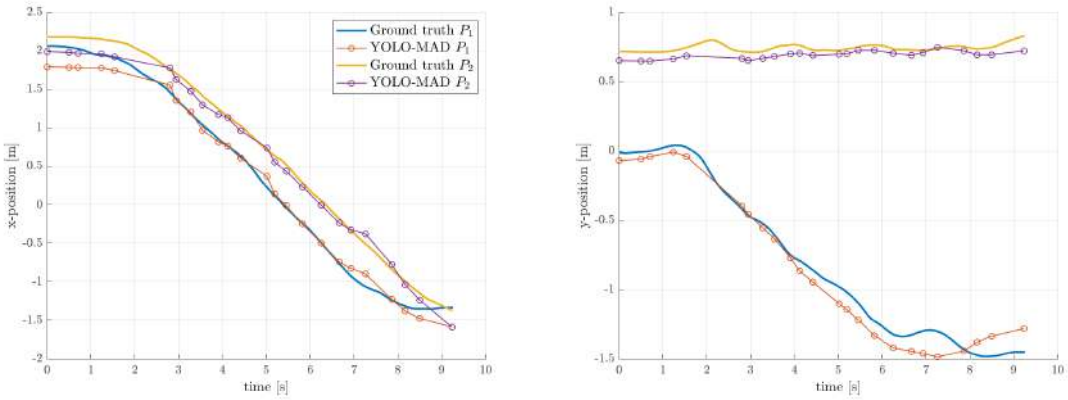
DOI: 10.13009/EUCASS2025-349



(a) x-position (b) x-velocity

(c) y-position (d) y-velocity

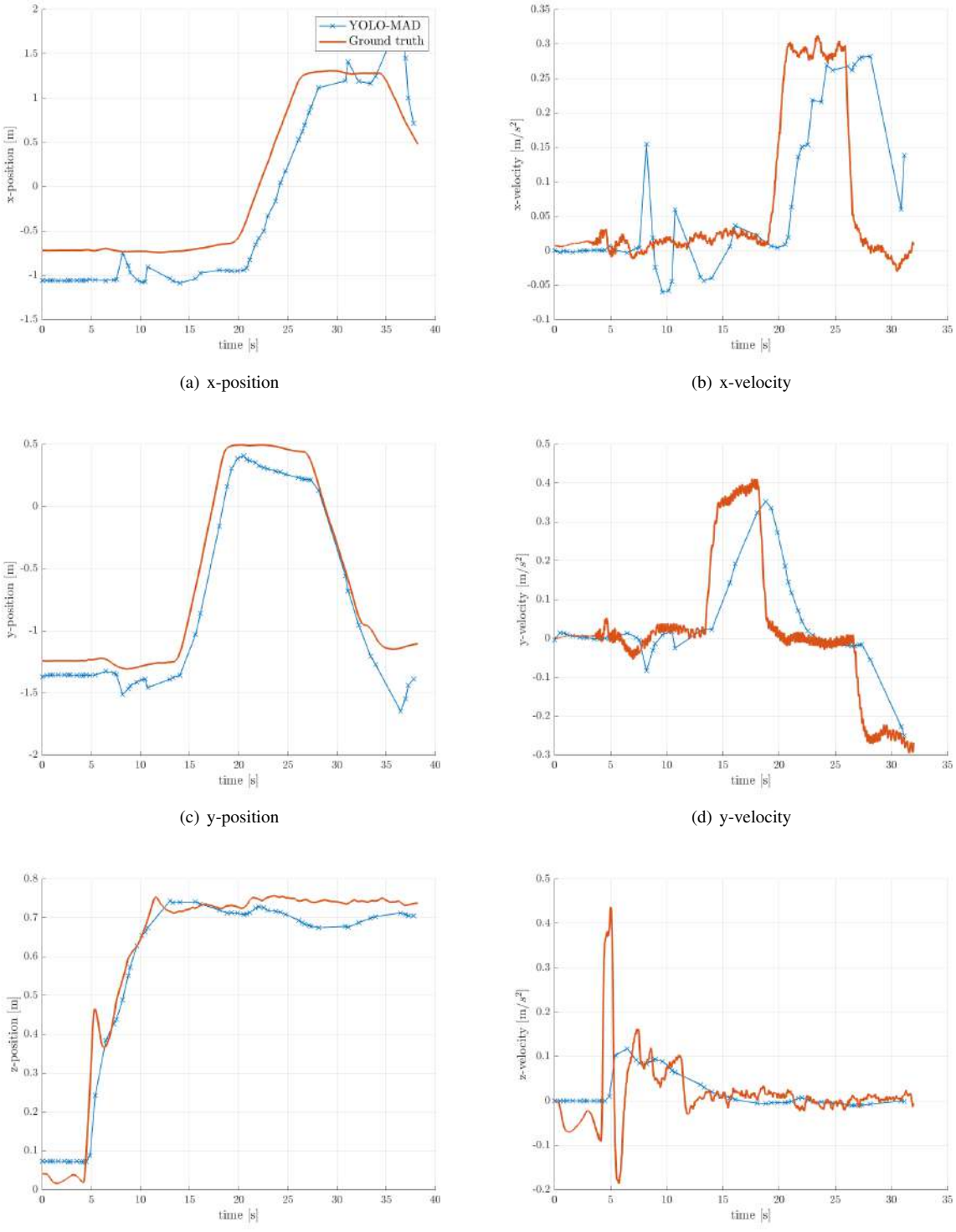
Figure 5: Exp #2: position and velocity estimated by YOLO-MAD and YOLO with U-Disparity.



(a) x-position (b) y-position

Figure 6: Exp #3: Ground truth and position estimated by YOLO-MAD.

DOI: 10.13009/EUCASS2025-349



(a) x-position

(b) x-velocity

(c) y-position

(d) y-velocity

(e) z-position

(f) z-velocity

Figure 7: Exp #4: Ground truth vs position and velocities estimated by YOLO-MAD.

DOI: 10.13009/EUCASS2025-349

Figure 8: In-flight drone detection

Conclusions

Recent advancements in drone technology have expanded the use of drones in various fields that require effective object detection. In this article, different algorithms for object detection using various data sources have been compared, evaluating their feasibility for use on drones. These algorithms include 3D methods, such as Cluster and U-Disparity, as well as the adoption of the YOLO neural network, which detects objects in 2D images. The proposed methods have demonstrated the capability of detecting objects in real-world scenarios. Moreover, we showed that using the data obtained from these methods it is possible to track objects' positions and estimate their velocity. We recorded an RMS of 0.21 meters (considering a range of distance in the experiment between 4 and 6 meters) when tracking a person. Instead, using a fine-tuned version of YOLO, we were able to obtain an RMS of 0.37 meters when tracking a drone. Additionally, the results indicate that YOLO significantly enhances the drone's detection capabilities, outperforming methods such as Clustering and improving the U-Disparity approach.

References

[1] R. Qi Charles, Hao Su, Mo Kaichun, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 77–85, Los Alamitos, CA, USA, July 2017. IEEE Computer Society.

[2] Naqqash Dilshad, JaeYoung Hwang, JaeSeung Song, and NakMyoung Sung. Applications and challenges in video surveillance via drone: A brief survey. pages 728–732, 10 2020.

[3] Anton Nazarov Dmitry Nazarov and Elena Kulikova. Drones in agriculture: Analysis of different countries. *BIO Web of Conferences*, 67, 09 2023.

[4] Thomas Eppenberger, Gianluca Cesari, Marcin Dymczyk, Roland Siegwart, and Renaud Dubé. Leveraging stereo-camera data for real-time dynamic obstacle detection and tracking. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, page 10528â10535. IEEE Press, 2020.

[5] Hossein Eskandaripour and Enkhsaikhan Boldsaikhan. Last-mile drone delivery: Past, present, and future. *Drones*, 7(2), 2023.

[6] Pedro F. Felzenszwalb, Ross B. Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010.

[7] Kuhn. *The Hungarian method for the assignment problem.* Naval Research Logistics Quarterly, 1955.

[8] Raphael Labayrade, Didier Aubert, and Jean-Philippe Tarel. Real time obstacle detection in stereovision on non flat road geometry through "v-disparity" representation. In *Intelligent Vehicle Symposium, 2002. IEEE*, volume 2, pages 646–651 vol.2, 2002.

[9] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. *SSD: Single Shot MultiBox Detector*, page 21â37. Springer International Publishing, 2016.

DOI: 10.13009/EUCASS2025-349

[10] Helen Oleynikova, Dominik Honegger, and Marc Pollefeys. Reactive avoidance using embedded stereo vision for mav flight. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 50–56, 2015.

[11] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection . In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, Los Alamitos, CA, USA, June 2016. IEEE Computer Society.

[12] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[13] Agoston Restas. Drone applications for supporting disaster management. *World Journal of Engineering and Technology*, 3(3):316–321, 2015.

[14] Arindam Saha, Bibhas Chandra Dhara, Saiyed Umer, Kulakov Yurii, Jazem Mutared Alanazi, and Ahmad Ali AlZubi. Efficient obstacle detection and tracking using rgb-d sensor data in dynamic environments for robotic applications. *Sensors*, 22(17), 2022.

[15] Hartmut Surmann, Artur Leinweber, Gerhard Senkowski, Julien Meine, and Dominik Slomma. Uavs and neural networks for search and rescue missions, 2023.

[16] Juan Terven, Diana Margarita Cordova Esparza, and Julio Alejandro Romero Gonzalez. A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas. *Machine Learning and Knowledge Extraction*, 5(4):1680–1716, 2023.

[17] Jesus Tordesillas and Jonathan P. How. Panther: Perception-aware trajectory planner in dynamic environments. *IEEE Access*, 10:22662â22677, 2022.

[18] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–I, 2001.

[19] Zhixin Wang and Kui Jia. Frustum convnet: Sliding frustums to aggregate local point-wise features for amodal 3d object detection. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1742–1749. IEEE, 2019.

[20] Zhefan Xu, Xiaoyang Zhan, Yumeng Xiu, Christopher Suzuki, and Kenji Shimada. Onboard dynamic-object detection and tracking for autonomous robot navigation with rgb-d camera. *IEEE Robotics and Automation Letters*, 9(1):651–658, 2023.