Real-time object geopositioning from monocular target detection/tracking for aerial cinematography

Daniel Aláez*, Vasileios Mygdalis†, Jesús Villadangos* and Ioannis Pitas†
*Mathematical Engineering and Computer Science Department, Public University of Navarre, Pamplona, Spain
† Department of Informatics, Aristotle University of Thessaloniki, Thessaloniki, Greece
Email:{daniel.alaez,jesusv}@unavarra.es,{mygdalisv,pitas}@csd.auth.gr

***Abstract*—In recent years, the field of automated aerial cinematography has seen a significant increase in demand for real-time 3D target geopositioning for motion and shot planning. To this end, many of the existing cinematography plans require the use of complex sensors that need to be equipped on the subject or rely on external motion systems. This work addresses this problem by combining monocular visual target detection and tracking with a simple ground intersection model. Under the assumption that the targets to be filmed typically stand on the ground, 3D target localization is achieved by estimating the direction and the norm of the look-at vector. The proposed algorithm employs an error estimation model that accounts for the error in detecting the bounding box, the height estimation errors, and the uncertainties of the pitch and yaw angles. This algorithm has been fully implemented in a heavy-lifting aerial cinematography hexacopter, and its performance has been evaluated through experimental flights. Results show that typical errors are within 5 meters of absolute distance and 3 degrees of angular error for distances to the target of around 100 meters.**

***Index Terms*—target detection, aerial cinematography, 3D geopositioning, monocular**

I. Introduction

Aerial cinematography has become an increasingly popular tool for capturing footage in a variety of media production contexts, including sports events, commercials, and movies. While manual operation offers a high degree of artistic control and the ability to achieve professional-looking shots, recent advancements in autonomous technology have made it possible to replicate these results through automated means. The literature has identified and described visually pleasing combinations of framing shot types and camera movements [1]. A common thread among these descriptions is the need for knowledge of the target's 3D position in a world coordinate system at all times. This requires the use of a target detection, localization, and tracking system, particularly in unstructured environments. It must be able to run in real-time, onboard the UAV, and should not rely on external sensors. Recent advances in computer vision techniques have emerged as a highly efficient means of performing object detection and tracking. By combining this technology with a pinhole camera model and a fast 3D position estimation algorithm, a highly efficient real-time object localization tool can be created for short and medium-distance UAV cinematography and aerial shot planning.

One of the most challenging aspects of aerial cinematography is subject localization for shot planning. Several works rely on motion-capture systems [2] or high-precision RTK GPS [3]. In cases where subjects cannot carry heavy or complex devices for tracking, the alternative must rely on visual image detection. Some typical scenarios are outdoor sports events (e.g. cycling and running) and search and rescue missions. Image detection algorithms have been extensively used for 3D position estimation in applications such as visual odometry of field robots [4], underwater vehicle positioning [5], and more recently in aerial cinematography [6].

A common approach to aerial cinematography using monocular cameras is based on visual servoing [7]. The main problem of visual servoing is that once the camera loses track of the subject, it is very challenging to recover. Therefore, a more robust alternative consists on ray casting, where we obtain 3D target positions that can be used to relocate it. Given the detection bounding box of the subject and using a ground model, the intersection of the ray cast from the camera to the bottom center of the bounding box with the ground model provides the 3D coordinates of the subject [6].

Nevertheless, ray casting requires pixel-accurate bounding boxes that are not easy to obtain. Modern state-of-the-art performing deep learning-based transformer-based detectors [8] or trackers [9], cannot be deployed in present onboard computing units. Besides detection and tracking, ray casting over complex ground models requires iterative solving, often unsuitable for real-time applications, since there is no analytical solution to the ray intersection with an arbitrary surface problem. Although the bibliography features the implementation of this system, we have not been able to find any information regarding accuracy, minimum and maximum usable distances, or even error estimation.

In this paper, we propose a real-time algorithm for estimating the 3D position of any object of interest based on image detection and tracking. By exploiting and combining the best practices in deep-learning-based visual detection and tracking, we are able to extract reliable target detections on the image plane in real time. Additionally, we provide a model for estimating the uncertainty in position prediction, considering factors such as bounding box selection, height, and pitch angle errors. To evaluate the performance of the proposed algorithm, we compare its results with actual data collected from experimental flights. The results demonstrate a high concordance between the estimated and actual positions.

II. METHODOLOGY

A. 2D Object detection and tracking

The first step of the proposed framework is to solve the problem of localizing targets of interest in the 2D camera coordinate system, in the form of rectangular Regions of Interest (ROI), over the course of successive video frames. Assuming that the filmed targets are standing still on the ground, we will employ the middle point of the bottom part of the bounding boxes (of the successive video frames), in order to cast rays toward the camera, whose position in the 3D space is known from other sensors. This Section focuses on how we extract those bounding boxes in real time.

Perhaps the most widely considered approach in embedded applications is the deployment of fast object detectors, such as the You Only Look Once (YOLO) series [10] or Single-Shot Detector (SSD) [11]. One of the most important limitations of detection-only approaches is they cannot easily generate between-frame correspondences, i.e., it is not trivial to deduct that the target detected in the previous frame is the same as the one depicted in the next. Next, detectors are heavily affected by environmental variance, related to camera motion, vibrations, illumination changes due to sun, etc. Detectors that are robust in such settings are transformer-based ones [8], however, they typically are challenging to implement in computationally and memory-constrained systems, such as drones. Another approach that solves the above issues is to employ visual object tracking methods. These methods are typically significantly faster than detectors that can run on CPU, e.g. correlation filter-based methods [12] or very fast on GPU, if they are based on siamese neural architectures [13]. Trackers are generally more adaptable to their environmental settings since they only learn to detect the target by using examples from previously tracked frames, which tend to be a lot more similar than a detection dataset where a detection method might have been trained on. The main limitation of tracking methods however is that they tend to be less accurate than detectors in the regression task, i.e., identify the exact proportions of the bounding box.

Inspired by [14], we have developed a complete Robot Operating System (ROS)-based software that combines the best of both worlds. The framework consists of 3 modules: a *detector*, a *tracker*, and an auxiliary *management* module. Operationally, the software works as follows: The management module handles input/output and triggers the detection and tracking modules. Given that there have been no previous detections, the management module provides images to the detection module. This operation is repeated until there are output detections. If a target has been detected, the management module uses the detected ROI to instantiate the tracking module. Unless a certain quality threshold is not achieved or a specific threshold of time (e.g., 25 frames) has been exceeded, the outputs of the software module are given by the tracker module. The tracker module always provides two outputs, one bounding box prediction, and one tracking quality score [15]. The acceptable quality threshold and the tracking time windows are the system's hyperparameters, which can be set prior to a filming mission. The management module takes into account the time and quality variables and decides whether to re-employ the tracking module or ask for new detections from the detection module. The methods used for the detection and tracking task are described in the following subsections.

1) 2D target detection: Object detection is the task of identifying and localizing object instances within an image, formulated as a combined classification and regression problem. Specifically, given an image sample $\mathbf{I}$, the detection model $\hat{\mathbf{y}} = \mathbf{f}(\mathbf{I}; \theta)$ learns a function parameterized by θ that predicts (assuming only one object instance) an output vector $\hat{\mathbf{y}} = [\hat{\mathbf{y}}_1^\top, \hat{\mathbf{y}}_2^\top]$ consisting of: (i) a class vector $\hat{\mathbf{y}}_1^\top \in [0,1]^m$, corresponding to a set of classes $\mathcal{C} = \{\mathcal{C}_i = 1, \ldots, m\}$ and (ii) a region of interest parameter vector $\hat{\mathbf{y}}_2 = [x, y, w, h]^\top$ defining the corresponding object's bounding box 2D coordinates.

Since the proposed framework does not require detections to take place in every single frame and is memory-allowed, we implemented a fast transformer-base detector [16], which is the state-of-the-art approach at the current period in general detection tasks. Such methods significantly outperform embedded-computing oriented detectors like Yolo and SSD in standard detection benchmarks, e.g., [17]. Inspired by the whitened self-attention operation [16], we developed a transformer-based detector that replaces the attention operation with a linear multiplication, by introducing auxiliary (pre-computed) matrices that perform a transformation that highlights known or computed data properties, modeled in graph structures. In particular, since we focus on detecting items that typically appear vertically elongated, we emphasize on attention operation along the y-axis. To this end, we employ a graph structure that connects vertical image patches with an increased weight. Since this property remains the same across any image patch, we can employ the same matrix for every possible given input image. Then, the attention operation is reformulated as follows [16]:

$$\mathbf{Z} = \sigma\left(\frac{\mathbf{X}\boldsymbol{\Sigma}^{-1}\mathbf{X}^T}{\sqrt{n}}\right)\mathbf{X}\mathbf{W}_V, \tag{1}$$

where n is a scaling parameter, $\boldsymbol{\Sigma} = \mathbf{X}^\top\mathbf{L}\mathbf{X} + r\mathbf{I}$ is matrix that encodes data relations that are expressed in a graph by the Laplacian matrix $\mathbf{L}, \mathbf{I}$ is an identity matrix of appropriate dimensions and $r > 0$ is a regularization parameter that adds a small value to the diagonal elements of $\boldsymbol{\Sigma}$ to ensure invertibility (a value of $r = 10^{-3}$ was used in out experiments). In our proposed framework, the same graph is employed for all elements, that connects nearby image regions with a fixed weight, i.e., a fixed Laplacian matrix. This allowed us to simplify the equation to:

$$\mathbf{Z} = \mathbf{A}\mathbf{X}\mathbf{W}_V, \tag{2}$$

where $\mathbf{A} = \sigma\left(\frac{\mathbf{L}^{-1}}{\sqrt{n}}\right)$ remains fixed during both training and inference stage. This reformulation remains easy to plug to standard Transformer-based neural networks, in a fully

end-to-end differentiable manner, being a transformer-based detection variant significanlty faster yet and more accurate than DETR [8]. The detection module implementing this method can output frames at a mean rate of 10 fps with standard *Jetson Xavier AGX* hardware, which is acceptable provided that the gaps in-between detections are filled with a tracking module. An example of a detection using this software is shown in Figure 1.

Fig. 1: Sample output of the proposed 2D detection and tracking software.

2) 2D target tracking: The target tracking is mathematically defined as the regression part of the detection task, i.e., $\hat{\mathbf{y}} = \mathbf{f}(\mathbf{I}; \theta)$, where $\hat{\mathbf{y}} = [x, y, w, h]^\top$ are the ROI coordinates of the target. The main assumption of tracking algorithms is that the tracked ROI always re-appears in successive frames, slightly translated, rotated and/or scaled. In aerial cinematography, however, besides estimating the affine transformation, another challenge is that the target also disappears from the field of view quite often. This is related to either abrupt camera/drone motions, that can introduce extreme noise due to vibrations (e.g., extensive blurring) or even completely lose the target. In fact, according to Visual Object Tracking Challenge reports [18], occlusions are the most common causes of tracking failure, hence they should be taken into account.

To address the above-mentioned challenges, we have opted for a two-fold approach. Since we only focus on finding the correct analogy of the bounding box, we have selected the SiamFC siamese tracker [13], which is very fast, and very good at maintaing and finding the correct scale of the bounding box. The algorithm was optimized for speed using the appropriate software acceleration libraries (i.e., TensorRT[1]). Furthermore, inspired by [15], we have developed a framework that accounts for target occlusions and estimates the tracking quality for each output bounding box. That is, instead of employing an SVM classifier, we trained a simple two-layer CNN network that works as a 3rd branch of the implemented tracker, that takes as input the detected features for selecting the ROI template and outputs a number between 0 and 1, where 0 means that the tracker believes that the target is lost. To train the algorithm, we have employed the VOT tracking benchmark, which is annotated for target occlusions. Thereby, the classifier was trained to detect occlusions from the tracker ouputs. In our experiments, we have set this parameter to 0.95, to only allow tracker outputs with high confidence.

[1]https://developer.nvidia.com/tensorrt

B. 3D Object geopositioning

Once the boundary box of the target has been identified relative to the center of the camera, the distance is obtained from the height of the drone above ground level (AGL) h, the angle of pitch of the camera θ, and the vertical angle between the rays that project to the camera focal center and to the middle bottom of the boundary box.

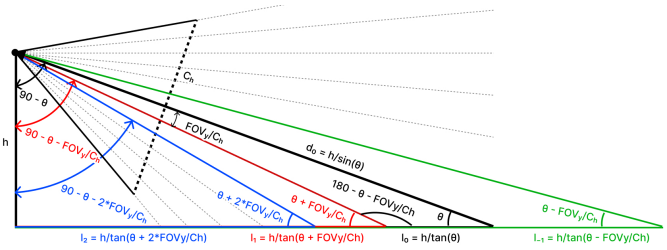

Fig. 2: Pinhole model and distance estimation diagram.

From the definitions in Figure 2, the distance l_i can be calculated as follows:

$$l_i = \frac{h}{\tan(\theta + i\frac{F_y}{C_h})}, \tag{3}$$

with $i \in \left[-\frac{\theta}{F_y/C_h}, \frac{C_h}{2}\right]$. The camera pitch ($\theta$) angle includes the fixed camera mounting angle and the instantaneous drone pitch provided by the flight controller. F_y and C_h are the vertical field of view of the camera and the height in pixels. The i and j distances from the center of the camera to the center of the bounding fox foot are given by $i = C_y - \frac{C_h}{2}, j = C_x - \frac{C_w}{2}$, where C_y, C_x C_h and C_w are the vertical center of the camera, the horizontal center, the height and width respectively. The point coordinates in the camera reference frame (X axis pointing forward to the normal of the camera plane and Y axis pointing right in the camera plane watched over the top) are $P_c = (l_i, l_i \tan \psi_{offset})$. To define the position in a North-East-Down (NED) coordinate system:

$$\begin{pmatrix} y \\ x \end{pmatrix}_{NED} = \begin{pmatrix} \cos\psi_{drone} & \sin\psi_{drone} \\ -\sin\psi_{drone} & \cos\psi_{drone} \end{pmatrix} \begin{pmatrix} y \\ x \end{pmatrix}_{C} \tag{4}$$

A very important property of the usual pinhole model for camera projection is that 3D lines in the scene are projected to 2D lines. Low-cost wide-angle lenses typically introduce a strong barrel distortion. For a fast and efficient lens distortion correction, a simple radial algebraic lens distortion model was used [19]:

$$\begin{pmatrix} \hat{x} - x_c \\ \hat{y} - y_c \end{pmatrix} = L(r) \begin{pmatrix} x - x_c \\ y - y_c \end{pmatrix}, \tag{5}$$

where $\hat{x}$, $\hat{y}$ are the corrected x and y coordinates and x_c, y_c are their respective image center coordinates. The lens distortion parameters are obtained by minimizing a 4 total-degree polynomial in several variables:

$$L(r) = k_0 + k_1 r + k_2 r^2 + k_3 r^3 + k_4 r^4, \tag{6}$$

where

$$r = \sqrt{(x - x_c)^2 + (y - y_c)^2}, \tag{7}$$

obtaining:

$$\begin{cases} k_0 = 9.593223e - 01 \\ k_1 = 0.000000e + 00 \\ k_2 = 5.338546e - 07 \\ k_3 = 0.000000e + 00 \\ k_4 = 1.671183e - 12 \end{cases} \tag{8}$$

for the selected camera. The bounding box camera coordinates are now suitable for 3D projection assuming the pinhole camera model.

1) Analyzed sources of error: From (3), we have identified the most relevant sources of error and combined the typical uncertainties for each variable based on observations and sensor manufacturer data:

- Height AGL (h): 2.5 m [20].
- Pitch angle (θ): 3.5 °.
- Bounding box (i): 30 px.

The uncertainty model, as a function of the three identified variables h. θ and i can be expressed by:

$$\delta d = \sqrt{\left(\frac{\delta h}{\tan(\theta + i\frac{F_y}{C_h})}\right)^2 + \left(\frac{-h\delta\theta}{\sin^2\theta + i\frac{F_y}{C_h}}\right)^2 + \left(\frac{-h\frac{F_y}{C_h}\delta i}{\sin^2\theta + i\frac{F_y}{C_h}}\right)^2} \tag{9}$$

with $i \in \left[-\frac{\theta}{F_y/C_h} + 1, \frac{C_h}{2}\right]$. Figure 3 shows the increase of the uncertainty in meters for each variable analyzed as distance increases. The bounding box and pitch uncertainties account for the largest growing error over distance, while the height estimation follows a more linear behavior for distances to target below 150 m.

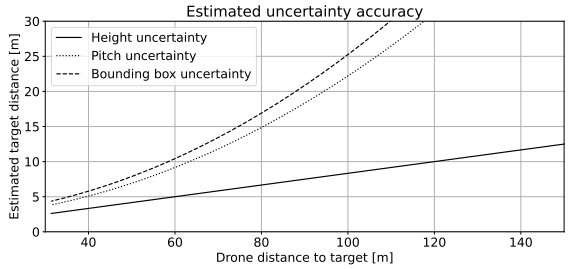

Fig. 3: Estimated uncertainty values for typical height, pitch angle and bounding box errors over distance to target.

III. Experiments and Results

To test the model, a flight experiment was carried out with a heavy-lifting aerial cinematography hexacopter in a safe open field environment. Figure 4 shows an overview of the hexacopter with the detection and tracking equipment installed. The location was chosen due to its safety and distance from populated areas. The key hardware components of the system are:

1) *Flight controller*: responsible for maintaining stable flight across the flight plan and sending real-time telemetry data including GPS and attitude via ROS. The flight controller software is based on Ardupilot.
2) *Positioning camera*: responsible for real-time image capturing and sending them to the onboard computer via ROS.
3) *Onboard computer*: Jetson AGX Xavier, responsible for receiving location, pose, and detection images, synchronizing times, and performing detection and tracking. The detection location is shared through ROS to coordinate aerial shots.

Object-of-interest locations can be shared via ROS to coordinate shots if multiple aerial cinematography drones are connected over the same network.

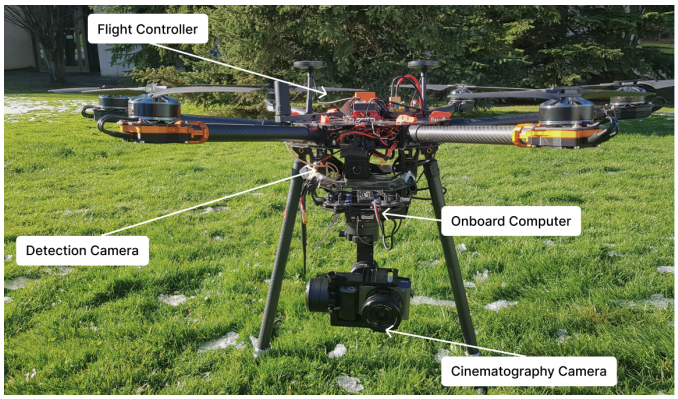

Fig. 4: Annotated photography of the hexacopter system architecture.

The selected objects of interest are electrical towers for their fixed and equally spaced positions. They also present features to guarantee the repeatability of the experiment. Nevertheless, this architecture is suitable for detecting all categories included in the COCO 2017 [21] pre-trained model without further training. The experiment conditions are listed below:

- Altitude: 30 m, constant AGL. Flight speed: 6 m/s.
- Parallel side-flight to the towers at a distance of 30 m.
- Number of towers flown for each pass: 4 towers.
- Camera mount angle: 15 ° of pitch to the bottom.
- Diagonal FOV: 92 °. Sensor type: 3:2. C_w: 640 px. C_h: 480 px.
- Magnetic declination at location: +0.76666 °.

The estimated angular error and distance error for each target are summarized in Figures 5 and 6, respectively. Additional video comparison with a YOLOv5 [22] detector and flight experiment videos are provided as supplementary material[2].

[2]This paper has supplementary video material available at https://youtu.be/NwakO8FnA5s and https://youtu.be/CJpHkhE2kSM, provided by the authors. This includes two videos featuring the 3D flight visualization with the detection overlay, and a live bounding box comparison of a standard YOLO detector and the DETR detector without tracking.

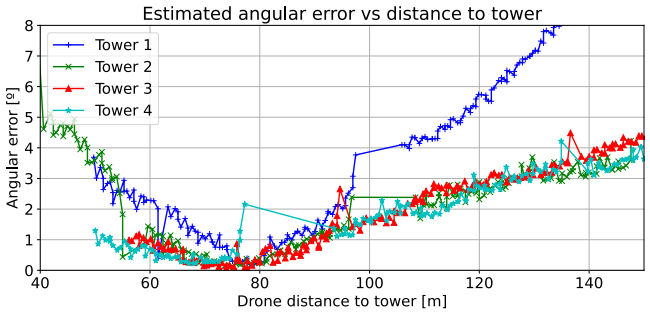

Fig. 5: Angular error in estimation over distance to subjects.

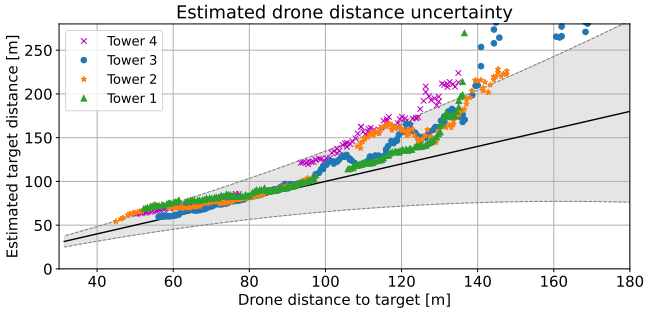

Fig. 6: Distance error estimation over actual distance to subjects.

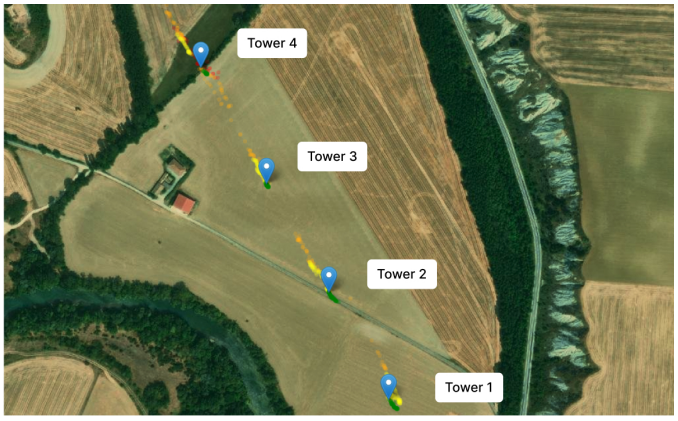

Fig. 7: Object of interest actual coordinates and algorithm-generated coordinates comparison.

The results demonstrate that distance and angular errors increase with distance to the target. This trend is more pronounced for distances greater than 100 meters. While the estimated distance values are well defined by the uncertainty model, errors are more significant than expected for long distances. This discrepancy is attributed to the implicit limitations of the flat terrain assumption used in the algorithm, as the experimental location featured slight uphill inclinations causing an overestimation of the distance.

In some cases, it is also observed how the error tends to increase at distances below 60 meters. Since the experiments were carried out parallel to the power line, when the drone is about to pass a tower (and thus the distance is minimum), the bottom of the tower falls out of the detection camera's field of view, but the algorithm is still able to detect it and track it. This produces inaccurate geopositioning of the tower. In future works we will consider identifying whether the detection of the tower may be incomplete to improve this aspect.

Figure 7 presents the experimental position estimations for each target on the map. The position estimation for objects closer than 120 meters approximately is reasonably good. This implies that the algorithm is suitable for most types of shots discussed in the bibliography [1].

IV. Conclusion

In this study, we have presented a real-time 3D position estimation algorithm for aerial cinematography based on image detection and tracking. The uncertainty of the algorithm was evaluated based on different variables, and the results were validated with experimental flight data. The results demonstrate a reasonable level of accuracy, with the vast majority of measurements below 100 meters of distance featuring an absolute error lower than 5 meters and 3 degrees of yaw. The algorithm's main limitation is using a flat-earth assumption for the ground model, which may be improved using an alternative ground model that does not require iterative solving. Future work includes implementing this algorithm in an automatic planning system for aerial cinematography shots.

Ackowledgement

This work has received funding from the European Union's European Union Horizon 2020 research and innovation program under grant agreements 951911 (AI4Media) and 871479 (AERIAL-CORE). This publication reflects only the authors' views. The European Commission is not responsible for any use that may be made of the information it contains.

References

[1] I. Mademlis, N. Nikolaidis, A. Tefas, I. Pitas, T. Wagner, and A. Messina, "Autonomous uav cinematography: A tutorial and a formalized shot-type taxonomy," *ACM Computing Surveys (CSUR)*, vol. 52, no. 5, pp. 1–33, 2019.

[2] Q. Galvane, J. Fleureau, F.-L. Tariolle, and P. Guillotel, "Automated cinematography with unmanned aerial vehicles," *arXiv preprint arXiv:1712.04353*, 2017.

[3] N. Joubert, D. B. Goldman, F. Berthouzoz, M. Roberts, J. A. Landay, P. Hanrahan *et al.*, "Towards a drone cinematographer: Guiding quadrotor cameras using visual composition principles," *arXiv preprint arXiv:1610.01691*, 2016.

[4] L. Piyathilaka and R. Munasinghe, "Multi-camera visual odometry for skid steered field robot," in *2010 Fifth International Conference on Information and Automation for Sustainability*. IEEE, 2010, pp. 189–194.

[5] S. He, Y. Liu, and J. Xiang, "A low cost visual positioning system for small scale tracking experiments on underwater vehicles," in *2020 IEEE 29th International Symposium on Industrial Electronics (ISIE)*. IEEE, 2020, pp. 1370–1375.

[6] R. Bonatti, W. Wang, C. Ho, A. Ahuja, M. Gschwindt, E. Camci, E. Kayacan, S. Choudhury, and S. Scherer, "Autonomous aerial cinematography in unstructured environments with learned artistic decision-making," *Journal of Field Robotics*, vol. 37, no. 4, pp. 606–641, 2020.

[7] J. Chen and D. M. Dawson, "Uav tracking with a monocular camera," in *Proceedings of the 45th IEEE Conference on Decision and Control*, 2006, pp. 3873–3878.

[8] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *European conference on computer vision*. Springer, 2020, pp. 213–229.

[9] X. Chen, B. Yan, J. Zhu, D. Wang, X. Yang, and H. Lu, "Transformer tracking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 8126–8135.

[10] R. M. Shah, B. Sainath, and A. Gupta, "Comparative performance study of cnn-based algorithms and yolo," in *2022 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, 2022, pp. 1–6.

[11] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[12] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 3, pp. 583–596, 2014.

[13] A. He, C. Luo, X. Tian, and W. Zeng, "A twofold siamese network for real-time object tracking," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4834–4843.

[14] P. Nousi, I. Mademlis, I. Karakostas, A. Tefas, and I. Pitas, "Embedded uav real-time visual object detection and tracking," in *2019 IEEE International Conference on Real-time Computing and Robotics (RCAR)*. IEEE, 2019, pp. 708–713.

[15] I. Karakostas, V. Mygdalis, A. Tefas, and I. Pitas, "Occlusion detection and drift-avoidance framework for 2d visual object tracking," *Signal Processing: Image Communication*, vol. 90, p. 116011, 2021.

[16] E. Patsiouras, V. Mygdalis, and I. Pitas, "Whitening transformation inspired self-attention for powerline element detection," in *2022 26th International Conference on Pattern Recognition (ICPR)*. IEEE, 2022, pp. 4844–4849.

[17] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, "Deformable {detr}: Deformable transformers for end-to-end object detection," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=gZ9hCDWe6ke

[18] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, J.-K. Kamarainen, H. J. Chang, M. Danelljan, L. Čehovin Zajc, A. Lukežič, O. Drbohlav, J. Bjorklund, Y. Zhang, Z. Zhang, S. Yan, W. Yang, D. Cai, C. Mayer, and G. Fernandez, "The tenth visual object tracking vot2022 challenge results," 2022.

[19] L. Alvarez, L. Gomez, and J. R. Sendra, "Algebraic Lens Distortion Model Estimation," *Image Processing On Line*, vol. 1, pp. 1–10, 2010, https://doi.org/10.5201/ipol.2010.ags-alde.

[20] "Here 3 manual," Jun 2022. [Online]. Available: https://docs.cubepilot.org/user-guides/here-3/here-3-manual

[21] H. Caesar, J. Uijlings, and V. Ferrari, "Coco-stuff: Thing and stuff classes in context," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1209–1218.

[22] G. Jocher, A. Chaurasia, A. Stoken, J. Borovec, NanoCode012, Y. Kwon, K. Michael, TaoXie, J. Fang, imyhxy, Lorna, Yifu), C. Wong, A. V, D. Montes, Z. Wang, C. Fati, J. Nadar, Laughing, UnglvKitDe, V. Sonck, tkianai, yxNONG, P. Skalski, A. Hogan, D. Nair, M. Strobel, and M. Jain, "ultralytics/yolov5: v7.0 - YOLOv5 SOTA Realtime Instance Segmentation," Nov. 2022. [Online]. Available: https://doi.org/10.5281/zenodo.7347926