

Enhancing Object Detection in UAV Imagery: A Modular Architecture For Object Permanence in Object Tracking Applications

Ekin Kağan Özkan

Defence Technologies Master Program

Istanbul Technical University

ozkan23@itu.edu.tr

514231009

I. ABSTRACT

Object permanence is a fundamental cognitive ability that enables humans to predict the locations of objects, even when they are not directly visible. In the field of computer vision, reproducing this capability has been a challenging task. While object tracking has seen significant advancements, there has been relatively little exploration of the object permanence problem. In this paper, we present a novel modular pipeline and model specifically designed to tackle the object permanence task. To enhance the performance of our approach, we integrate a monocular depth extraction module into the pipeline. Our proposed method is trained and evaluated on the CATER dataset, demonstrating its effectiveness for the object permanence task.

II. INTRODUCTION

Good object tracking is an essential part for many applications. It has many different approaches in the literature. Though many of them are lacking the tracking of non visible objects. This task is called object permanence in human psychology field. [add some info about object permanence here, piaget, infants]. In ... work they divide the op task into 4 different subtasks. These four tasks are (1) visible, (2) occluded, (3) contained or (4) carried. In our work we combined contained and carried subtasks into contained task. Since they are same tasks in the perspective of a current moment.

However, the effectiveness of these UAV-based object detection systems is very often compromised by the adverse weather conditions of fog and haze, which obscure visual details and reduce the clarity of the captured images.

Fog and haze are typical atmospheric phenomena that scatter the light; due to this, they create lousy visibility conditions, and standard object detection is quite challenging. This problem is raised while dealing with aerial images, where a great distance between the camera and objects of interest can underline the impact of these weather conditions. Therefore, further development of effective object detection models that will work well in this condition is needed. The YOLO, which stands for "You Only Look Once," family of models, has

had a tremendous acceptance with tasks involving real-time object detection and remarkable accuracy in various computer vision tasks. Its most recent member, YOLOv9[1], promises improvements in inference speed and accuracy. Still, for a considerable percentage of time, the performance of YOLOv9 remains abysmal under fog and haze conditions, similar to many other object detection models.

This limitation will be addressed with this project by fine-tuning the YOLOv9 model to enhance its object detection capability on images taken under hazy and foggy conditions.

We developed two different datasets for this purpose. The first dataset is synthetic; applying a customized fog effect on clear images was accomplished, using a slight modification of the general technique that simulates realistic fog conditions in images. Now, the amount of fog can be controlled, and a wide variety of visibility conditions can be achieved using this method.

The second dataset consists of real-world images scraped from YouTube, with drone footage taken under different hazy and foggy conditions. These tend to present a wide variety of real-world settings and features, making the valuable dataset in the evaluation of the modeled realistic settings. The collection from publicly available videos is supposed to make sure that the dataset covers a broad spectrum of weather conditions and types of objects for overall generalizability.

This paper presents how the datasets were created initially and the fine-tuning of the YOLOv9 model in correspondence with these datasets. It further discusses data fine-tuning, an implemented process whereby a pre-trained YOLOv9 model is retrained using the presented datasets to suit the current task and its requirements of object detection. During training, we used some data augmentation techniques to improve the robustness of our model. Augmented data allows the model to learn to detect objects in different degrees of visibility and other environmental conditions.

The performance of the fine-tuned YOLOv9 model was estimated through a series of metrics, including precision, recall, and mean Average Precision (mAP). The results indicate that the fine-tuned model significantly outperforms the baseline

YOLOv9 model in detecting objects under hazy and foggy conditions. Thus, the enhanced model shows good potential for application in real-world problems with poor weather as a norm. This project contributes to the field of computer vision by addressing the following anomaly: object detection in degraded visual environments. We fine-tuned the YOLOv9 model in a transfer learning setup with careful curation of synthetic and natural datasets to address this anomaly, thereby developing a more resilient and accurate detection system for UAV imagery. This paper opens the way toward additional innovation and improvement in the range of detection of objects under bad weather conditions and other additional scenarios given reaching more reliable and effective UAV-based systems in broader applications.

III. METHODOLOGY

A. Data Collection

To address the challenges of detecting objects in hazy and foggy UAV images, we developed two distinct datasets:

Synthetic Dataset: This dataset was created by applying a custom fog effect to clear images. The fog effect was generated by modifying the Fohis technique, which is known for simulating realistic fog conditions. This allowed us to control the density and distribution of the fog, ensuring a diverse range of visibility conditions. The synthetic dataset includes various object types and scenes to provide a comprehensive training set for the model.

For this dataset we combined two datasets that are available for public use, VisDrone Dataset[2] and VSAI Dataset[3]. After researching the topics of haze and fog simulation in UAV images, as well as dehazing and defogging operations on UAV images, we observed a lack of UAV image-focused approaches for fog and haze simulation. Additionally, there is a scarcity of low-cost solutions for dehazing and defogging operations for UAV images. Therefore, we propose UAV-focused approaches for both of these topics.

Real-World Dataset: To capture real-world scenarios, we scraped images from YouTube videos featuring drone footage in hazy and foggy conditions. These images encompass a wide variety of environments and levels of fog density, offering valuable real-world examples for training and evaluation. And merged this dataset with the Mafat-Haze[4] dataset. The diversity in this dataset ensures that the model is exposed to various fog conditions and object types.

B. Data Preprocessing

1) Homogeneous and Heterogeneous Fog and Haze Simulation on Images Taken from UAVs: To simulate the homogeneous fog in our input image, we must create a random Perlin noise with Gaussian blur. With this, we will be able to simulate the fog and haze effect more randomly and independently from depth and perspective data. Normally, FoHIS [5] creates the result image in two steps: homogeneous and heterogeneous fog and haze simulation.

However, in our situation, since we are using the randomized 2D Perlin noise with Gaussian blur that we generated,

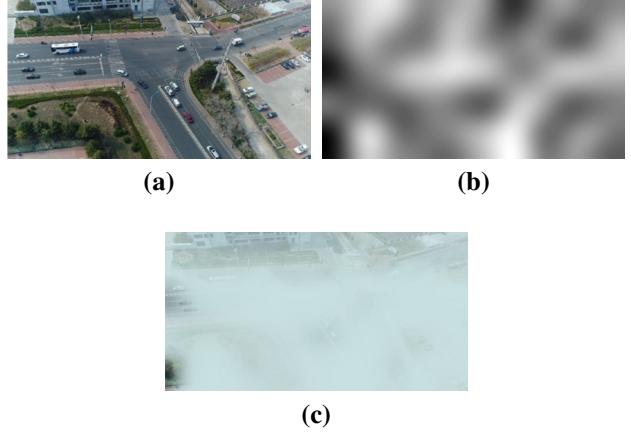


Fig. 1. Haze and fog simulation using randomly generated 2D Perlin Noise with Gaussian Blur. (a) original image. (b) randomly generated 2D Perlin Noise with Gaussian blur. (c) final result from haze and fog simulation operation.

we are creating the result in the first step already. Because of the Perlin noise's random and heterogeneous nature, we are fulfilling the heterogeneous part, and with the blurring and evenly distributing effect of the Gaussian blur, we are fulfilling the homogeneous part.

So, to generate the noise, we first need Perlin noise.

$$\text{PerlinNoise}(x, y) = \sum_{i=0}^{2^n-1} \sum_{j=0}^{2^n-1} \text{grad}(x_i, y_j) \cdot \text{Fade}\left(\frac{x - x_i}{2^n}\right) \cdot \text{Fade}\left(\frac{y - y_j}{2^n}\right) \quad (1)$$

with Gaussian blur,

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

and as a final operation,

$$\begin{aligned} \text{BlurredPerlin}(x, y) &= \sum_{i=-k}^k \sum_{j=-k}^k G(i, j) \cdot \text{PerlinNoise}(x - i, y - j) \\ &= \sum_{i=-k}^k \sum_{j=-k}^k \frac{1}{2\pi\sigma^2} e^{-\frac{i^2+j^2}{2\sigma^2}} \cdot \text{PerlinNoise}(x - i, y - j) \end{aligned} \quad (2)$$

We generated our Gaussian-blurred 2D Perlin noise. Figure 1 shows the input image, our generated 2D Perlin noise, and the result image from this haze and fog simulation operation. Some parts of the image are slightly foggy and hazy due to our Gaussian blur, while other parts of the image are heavily foggy and hazy as a result of the solid parts of our 2D randomly generated Perlin noise.



Fig. 2. put of the hazy and foggy image

2) Dehazing and Defogging of UAV Images: We modified the He et als.[6] solution for the UAV images. To create the dark channel prior map, we must get the lowest value of every pixel's every channel. Then, we create a map based on the lowest value of each pixel. So for an image J , we define:

$$\text{Dark Channel Prior}(J) = \min_{c \in \{R,G,B\}} \left(\min_{(i,j) \in \Omega(i,j)} J^c(i,j) \right)$$

where $J^c(i,j)$ represents the intensity of the pixel at position (i,j) in channel c , and $\Omega(i,j)$ is the local patch centered at (i,j) .

Normally this technique is used to define where the sky is located in the image and where the other objects are located. Because for the sky region, the value of J_{dark} is higher than zero, and for most of the non-sky regions, J_{dark} is low or equals to zero. We use this property to map the fog and haze in the image, since foggy and hazy parts are going to be more brighter, their J_{dark} values are going to be higher, and non-foggy and non-hazy parts' J_{dark} values are going to be lower or zero. Figure 2 shows the output from our dark channel mapping operation.

In our operation we are going to use patches to calculate the J_{dark} values,

$$J_{\text{dark}}(x) = \min_{c \in \{r,g,b\}} \left(\min_{y \in \Omega(x)} J^c(y) \right) \quad (5)$$

Which J^c represents the intensity data of channel c (red, green or blue) of the pixel y , and $\Omega(x)$ denotes the local patch that centered at x , with this operation we will be able to find the minimum intensity of each pixel that is in the patch. Then we are going to estimate the atmospheric light in our image using,

$$A = \max_x (J_{\text{dark}}(x)) \quad (6)$$

where A is the estimated atmospheric light, often approximated as the whitest (brightest) pixel in our dark channel, we assume it is the least affected pixel from haze and fog. After estimating the atmospheric light, we are going to calculate the transmission map with that using,

$$t(x) = 1 - \min_{c \in \{r,g,b\}} \left(\min_{y \in \Omega(x)} \left(\frac{J^c(y)}{A^c} \right) \right) \quad (7)$$

which $I^c(y)$ represents the intensity values of color channel c and A^c represents the atmospheric light value for channel c . And as a last step, we are going to calculate the dehazed and defogged image with,

$$J(x) = \frac{I(x) - A}{\max(t(x), t_0)} + A \quad (8)$$

as $J(x)$ is our dehazed and defogged image and t_0 is the small constant to prevent division with zero.

3) Labeling the Images for YOLO Fine-Tuning Operation: To train the YOLOv9 model accurately, it should be labeled properly. This is how we carried out the image labeling:

Software Selection: We employed RoboFlow, a straightforward, image labeling web tool. Data can be retrieved via api to Google Colab, making it adequate, compatible, and valuable for our project.

Annotation Steps: For each image in both synthetic and real-world datasets, we performed the following steps:

- 1) **Images Loading:** We loaded every one of the images into RoboFlow.
- 2) **Draw Bounding Boxes:** Using an image, we created a box for each object.
- 3) **Labeling:** We annotated each box with the correct class, for example, car, house, and person.

YOLO Format: RoboFlow saves the annotations in YOLO format and includes:

- **Class ID:** A number representing the class of an object.
- **Bounding Box Coordinates:** Normalized center x, y, width, and height of the box.

Quality Checks to Ensure Accuracy:

- **Human Review:** A second reviewer independently reviewed each labeled image.
- **Corrections:** All errors were corrected.
- **Consistency:** We checked for consistent labeling across all images.

MODEL FINE-TUNING

The fine-tuning process of the YOLOv9 model—detection of objects in the hazy and foggy UAV images—is achieved by training two separate models using two datasets: a synthetic dataset and a real-world dataset. The procedure is explained further in the next section.

1- Initial Setup

Transfer Learning: Our starting point was the pre-trained YOLOv9 model, which has been trained on the COCO dataset. This is a solid foundation because the model is already well-informed about general object detection.

Environment Setup: The training environment was set up to use high-capability GPUs. Libraries and frameworks required to be installed and configured include PyTorch using Google's Colab service.

2- Training the Model

Tuning the Hyperparameters: The following major hyperparameters were tuned in training for optimization:

- **Learning Rate:** Regulated how much the model's weights are updated with each training step. A learning rate scheduler was used to adapt the rate dynamically.
- **Batch Size:** The number of images processed at each training step, balanced to make good use of GPU memory without slowing the training speed.
- **Epochs:** The number of complete passes through the training dataset, adjusted to avoid overfitting.
- **Loss Function:** YOLOv9 loss is a mixture of three parts:
 - **Classification Loss:** Evaluates how good class probability predictions are compared to the ground truth.
 - **Localization Loss:** Measures how well the predicted bounding boxes are accurate.
 - **Confidence Loss:** Measures how accurate the objectness score is that states whether an object exists in the bounding box.

Training: Each model was tuned and trained using the corresponding datasets:

- **Model of Synthetic Dataset:** Training of the model was done on the synthetic dataset, which included images that spanned from none to high levels of fog applied by our custom fog effect.
- **Real-World and Synthetic Dataset Model:** Training was performed on the real-world dataset of hazy and foggy images and synthetic dataset mixed.

During training, the following steps were performed:

- **Forward Pass:** An input image passes through the model to predict something.
- **Total Loss Calculation:** The loss was an addition of classification, localization, and confidence losses.
- **Backward Pass:** From the gradients, the model weights are updated to minimize the loss.

4. Validation and Evaluation

Validation at Training Time: The performance of each model was checked on the validation set at the end of an epoch so far trained. It was helpful to monitor overfitting and adjust the learning rate or other parameters as appropriate.

Final Testing: Both models have been evaluated over several metrics after training:

- **Precision Calculation:** The number of true positive detections out of the positive detections taken up by the model.
- **Recall:** The ratio of accurate positive detections to the overall instances in the dataset that are positive. It is calculated by considering the mean average precision (mAP) of both precision and recall over various detection thresholds, giving a measure covering the overall performance of a model.
- **Visualization of the Results:** The results were visualized using bounding boxes on the validation images to qualitatively assess each model's performance.

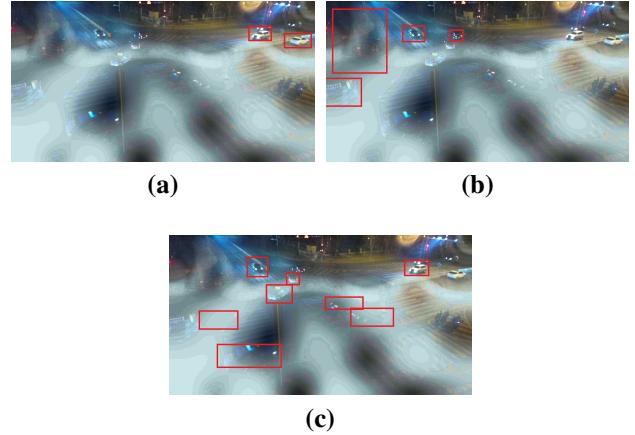


Fig. 3. Result images from our models (a) Base YOLOv9 Model. (b) Model that is trained with real dataset. (c) Model that is trained with synthetic and real dataset.

C. Results

The performance of the YOLOv9 model in detecting objects in hazy and foggy UAV images was evaluated using three different configurations: base, fine-tuned with non-synthetic, and fine-tuned with a combined dataset of synthetic and non-synthetic images.

Model Name	Precision	Recall	mAP50
Model trained with real data	0.82	0.564	0.662
Model trained with mixed data	0.859	0.109	0.0908

TABLE I

VALIDATION RESULTS OF MODELS TRAINED WITH DIFFERENT DATA TYPES

Base YOLOv9 Model: The base YOLOv9 model, pre-trained on the COCO dataset, is run on hazy and foggy UAV images without any fine-tuning. This model attains an precision of 5.6%, which is very poor, indicating that the model has minimal capability for object detection under adverse weather conditions. It needs special training to handle foggy and hazy scenarios unless indicated by poor performance.

Fine-Tuned YOLOv9 Model with Non-Synthetic Dataset: The first fine-tuning approach was made by taking the YOLOv9 model and then training it on non-synthetic data collected from real-world hazy and foggy images scraped from YouTube. This model achieved 48% precision, indicating a massive improvement in performance. The significant increase in precision demonstrates the effectiveness of using real-world samples, which help the model learn to detect objects under deteriorated visual conditions.

Combined Dataset: Fine-Tuned YOLOv9 Model with Synthetic and Non-Synthetic Combined Dataset: The other approach involved fine-tuning the model with a combined dataset of synthetic and real-world images. This model yielded up to 57% precision. The inclusion of synthetic images with influenced fogging exposure provided greater variability during training, making the model more robust and able to detect objects in a variety of foggy and hazy conditions.

Summary of Results: While the base YOLOv9 model had 5.6% precision, the fine-tuned model with the non-synthetic dataset achieved 48% accuracy, and the one fine-tuned with the combined dataset achieved 57% precision. These results indicate that fine-tuning the YOLOv9 model vastly improved its performance in object detection through UAV images taken in hazy or foggy conditions. The best performance was obtained from the model trained on the combined dataset, proving the importance of training with a mixed dataset of real-world and synthetic data. This approach enables the model to generalize better and perform effectively in adverse weather conditions.

These findings highlight the potential of fine-tuning pre-trained models on specialized datasets to address specific challenges within the object detection task. Future work could extend the integration of more data augmentation techniques and synthetic effects to further diversify the training data, which might result in additional improvements in detection precision in challenging environments.

REFERENCES

- [1] C. Wang, I. Yeh, and H. M. Liao, "YOLOV9: Learning what you want to learn using programmable gradient information," *arXiv.org*, Feb. 21, 2024. [Online]. Available: <https://arxiv.org/abs/2402.13616>.
- [2] P. Zhu, L. Wen, D. Du, X. Bian, H. Fan, Q. Hu, and H. Ling, "Detection and tracking meet drones challenge," *IEEE Trans. Pattern Anal. Mach. Intell.*., vol. 44, no. 11, pp. 7380-7399, 2021.
- [3] J. Wang, X. Teng, Z. Li, Q. Yu, Y. Bian, and J. Wei, "VSAI: A Multi-View Dataset for Vehicle Detection in Complex Scenarios Using Aerial Images," *Drones*, vol. 6, no. 7, p. 161, 2022. doi: 10.3390/drones6070161.
- [4] "mafah-haze Dataset," *Roboflow Universe*, Apr. 2022. [Online]. Available: <https://universe.roboflow.com/nguyennguyen4601-gmail-com/mafah-haze>. [Accessed: Jun. 7, 2024].
- [5] N. Zhang, L. Zhang, and Z. Cheng, "Towards simulating foggy and hazy images and evaluating their authenticity," in *Proc. Int. Conf. Neural Inf. Process.*., 2017, pp. 405-415.
- [6] He, K., Sun, J., Tang, X.: Single image haze removal using dark channel prior. IEEE Trans. PAMI 33(12), 2341–2353 (2011)