Eye in the Sky: Tracking a Moving Animal Using Deep and Traditional Methods
A Comparison of Deep Learning and Traditional Object Detection in Simulated Environments

Paul Bliemegger
Justin Wenzel

Team Number: 8

March 12, 2025

Project Proposal for
HCI575: Computational Perception

Iowa State University
Spring 2025

ABSTRACT

Object detection and tracking are popular topics in computer vision and are present in many applications, including surveillance, self-driving cars, and robots. With many approaches being proposed in these areas, the idea for more comparative studies of different situations opens up. This project aims to compare and evaluate traditional and deep-learning-based approaches. For object detection, the model YOLO (deep learning) is compared against HOG+SVM (traditional), while Deep-SORT (deep learning) is compared with a Kalman Filter (traditional) approach for tracking.

To perform this comparison, we utilize a controlled experimental setup using Microsoft Airsim, where a drone-mounted camera will scan and track a virtual dog that is walking. Performance will be measured under different conditions, including a basic motion path, a path deviation, and occlusion. Multiple factors are monitored, including detection accuracy, tracking stability, and computational efficiency. This project provides a comparative analysis of existing techniques in different simulated scenarios, providing insight into each techniques' performance in different scenarios.

1 Introduction

1.1 Problem Statement

Object detection and tracking are a continuing problem in computer vision tasks with a growing interest in critical applications such as surveillance, self-driving cars, and robots. Researchers have proposed various techniques to solve this continuing problem, ranging from traditional machine learning-based methods such as Histogram of Oriented Gradients (HOG) + Support Vector Machines (SVM) to deep learning models such as You Only Look Once (YOLO) for object detection. Tracking methods also include various techniques, including mathematical methods such as the Kalman Filter and deep learning-based models such as DeepSORT. Deep learning-based approaches have been a major shift in computer vision performance compared to traditional approaches but still have limitations due to their high computational complexity and processing speeds [1]. A major challenge involved in assessing different object detection and tracking techniques is testing how they behave under different conditions. Common research approaches involve using real-world datasets, but simulation testing can have benefits also which include controlled variations, reproducibility, and low cost. Microsoft Airsim is a high-fidelity simulation platform that offers a systematic and reproducible environment for testing detection and tracking approaches in a dynamic environment [2]. This is particularly useful for analyzing different object detection and tracking approaches on varying objects, such as animals whose motion is less predictable than a human or car. As the quality of detection plays a critical role in tracking results, it is important to study how different detection and tracking are influenced in diverse scenarios.

1.2 Problem Motivation

While deep learning-based models have shown success in computer vision tasks, most existing work focuses on evaluating the performance of these proposed ideas against the weaknesses of other approaches in specific tasks. However, real-world accurate detection and tracking is a complex problem involving non-rigid objects such as animals, irregularly moving objects, and a dynamic environment, which are all essential nonspecific tasks for critical computer vision tasks. As a result there is no object detection or tracking approach that is superior in every circumstance, and often, performance is highly sensitive to environmental conditions, complexity of motion, and occlusions. Additionally, real-world testing can be costly and difficult to control specific scenarios, making simulation-based testing a suitable alternative. Microsoft AirSim allows researchers to simulate real-world environments and test different approaches with various conditions before real-world deployment. Testing detection and tracking approaches in a simulated environment provide valuable insights into the approach's performance against dynamic scenarios.

A side affect of our method is experimenting with using simulated data for machine learning models. As data is very important in the current world of machine learning, knowing how well simulated data can be used in these models, might help in the process of finding good quality data. Finding useful data is not trivial. [3]

1.3 Research Objectives

This projects objective is to compare and evaluate classical and deep-learning-based approaches for object detection and tracking in a simulated controlled environment. More precisely this research

aims to compare YOLO and HOG+SVM for object detection, and DeepSORT and Kalman Filter for object tracking. The operation of these techniques will be compared based on the accuracy of how well objects are identified, tracking stability of the identical object across frames, and computational efficiency regarding various patterns of motion, such as a straightforward trajectory of motion, a path deviation such as zig-zag, and occlusion of an object. This project can provide an overview of different object detection and tracking approaches to enhance the knowledge of performance against the same scenarios with different levels of complexity that could map to other potential real-world applications.

2 Related Work

Object detection and tracking have been a long-time problem in computer vision and, to this day, range in complexity from hand-crafted feature extraction to complex deep learning models. A popular classical approach HOG+SVM began because of original feature extraction work known as Histogram of Oriented Gradients (HOG) introduced by Dalal & Triggs [4], which began a foundation for object detection with its combination of involving State Vector Machines. While compared to this traditional method, deep learning-based detectors such as YOLO found by Redmon, Divvala, Girshick, and Farhadi [5] have continued to evolve over time with researchers developing different variants of the model, including YOLO9000 [6] and YOLOV3 [7]. For object detection and tracking to work together, tracking algorithms must evolve over time also and research has continued with the same advancements as object detection, beginning with traditional methods such as the Kalman Filter introduced by Kalman [8]. The original Kalman filter has evolved over time and is now also used with DeepSORT in a deep learning approach for real-time object tracking that also evolved from SORT, a simple online real-time tracking approach proposed by Bewley, Paulus, and Wojke [9]. These deep learning methods are often found together and have been tailored for many applications, leading them to be insightful techniques to explore. A group of researchers also proposed a deep learning-based object tracking model that they compared and analyzed trade-offs with other approaches, including DeepSORT [10]. These research papers all show the importance of evolving approaches in object tracking and detection, along with the importance of constantly evaluating and comparing the approaches with each other to help determine strengths and weaknesses for each approach in different use cases.

Microsoft Airsim has been a widely used simulator for testing deep learning-based computer vision applications in vehicle and aerial robotics. Research conducted by Benoit, Xing, and Tsourdos used AirSim to generate synthetic data for a YOLO model to detect long distance airborne objects with great results [11]. While another group of researchers from ELAN Microelectronics Corporation and the National Taiwan University used AirSim for traffic object detection in virtual environments to simulate a real street view again using YOLO models for object detection [12]. While our work is not with synthetic data specifically or object detection from a vehicle, this research highlights Airsim's potential in our research to provide a controlled and consistent environment for analyzing real-time detection and tracking methods.

As we are dealing with object detection in a simulated environment, the article "Development of a Novel Object Detection System Based on Synthetic Data Generated from Unreal Game Engine"[13] falls into a similar category. Here, they explored the idea of creating synthetic training data for deep learning models from scenes in Unreal Engine 4 and apply these models on real-world

data. This is supposed to simplify the model training process. While our goals differ, their testing of models trained on simulated data will definitely help us in adjusting and fine tuning ours, as well as give us some possible expectations for our results. They also provide possible improvements which we will take into account for our project. Another key takeaway from their research is how our approach of using simulated environments for object detection has various practical use cases. To further proof this point, comparisons of other approaches also seem viable as with agent based tracking which this article [14] used to compare the performance of Deep-Q-Networks and Proximity Policy Optimization object tracking agents.

3 Experimental Platform

For the control of the drone as well as acquiring images and footage for our further comparison processes we are using Microsoft AirSim. This open-source simulator provides an extensive API with a detailed documentation. Although further support has been deprecated, it was released in a stable state. Microsoft AirSim also provides implementations for Unity and Unreal Engine 4. We decided on the latter, as the Unity implementation is in an experimental state. Access to the API is easily possible via Python. This allows us to take advantage of all the drone's camera sensors (Lidar, Infrared and distance), as well as to its flight controller. While there is the option to define or use more powerful controllers, AirSim provides a "simple controller" which already provides a lot of funtionality out of the box. After retrieving footage from the sensors via the API we can then use OpenCV for image pre- and post-processing. With this we are aiming to achieve fast speeds that allow us to keep real-time object tracking.

As we are using Unreal Engine 4 as the base for our simulation, we are able to use all the functionalities of a commercially viable real-time game engine as our playground. This allows us to create complex scenes and scenarios without the worry of hits in performance. We can also use the many models available for such game engines, increasing the variety of possible scenarios.

Figure 1: Using Microsoft AirSim to place a drone in a complex environment

Figure 2: Footage of the drone camera retrieved using Python and OpenCV

4 Methodology

4.1 Data Collection in Simulated Setup

Carrying out the simulation in a virtual environment with the help of Microsoft Airsim enables easier implementation and extraction of results, where a drone captures a real-time feed of a moving object. The object of interest being tracked in this instance is a virtual dog with prescripted movement patterns; then, various object detection and tracking approaches will be used to test their performance against the various movement patterns.

Captured frames using the AirSim API are pre-processed to ensure they are compatible with the selected models. Such preprocessing operations can consist of resizing, normalization, and conversion to grayscale depending on what is required for the object detection and tracking technique as an expected input format.

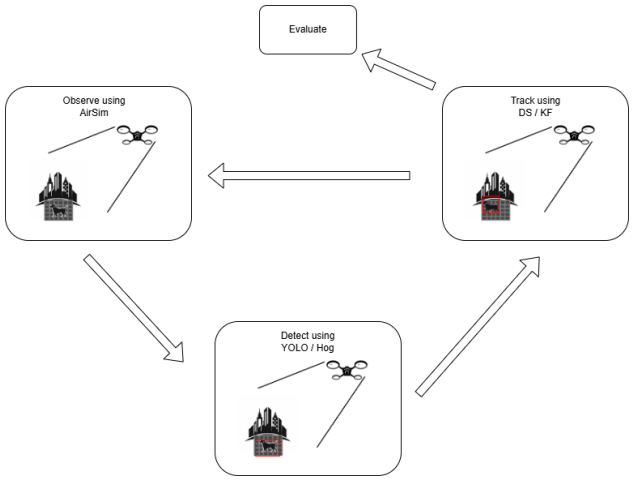

Figure 3: Simple overview of methodology

4.2 Object Detection Approaches

For object detection, two approaches are compared:

- **YOLO (You Only Look Once)** [5] – A deep learning-based detection model known for high accuracy and real-time performance.
- **HOG + SVM (Histogram of Oriented Gradients with Support Vector Machine)** [15] – A traditional feature-based approach that detects objects based on edge structures.

4.3 Object Tracking Approaches

After detection is performed, tracking algorithms are used to maintain object identity across frames:

- **DeepSORT** [16] – A deep learning based tracker that uses motion and appearance information to maintain object identity consistently.

- **Kalman Filter** [17] – A traditional approach that predicts object location over time based on past estimates.

4.4 Experimental Scenarios

To examine the robustness of these detection and tracking approaches, the combined techniques will be conducted in different three scenarios:

- **Basic Motion Path**: Object moves in a straight-line trajectory.
- **Zig-Zag Motion**: Object moves with unpredictable turns.
- **Occlusion Scenarios**: Object is temporarily blocked by obstacles.

4.5 Performance Evaluation Metrics

The robustness of these object detection and tracking approaches will be assessed based on three primary evaluation metrics:

- **Detection Accuracy** – The percentage of correctly detected objects compared to ground truth annotations.
- **Tracking Stability** – The ability of the tracking method to maintain a consistent object ID across frames, even in occlusion scenarios.
- **Computational Efficiency** – Measured in frames per second (FPS) and processing time per frame to determine the feasibility of real-time applications.

4.6 Challenges and Limitations

While using a simulated platform such as Microsoft Airsim to assist in experiments provides a controlled and reproducible environment for evaluating object detection and tracking approaches, there are a few limitations to be considered when interpreting the results. A primary concern is the inconsistency between synthetic and real performance as simulated worlds do not represent true lighting conditions, texture variations, or sensor noise, which all can affect the ability for deep learning models to generalize and traditional approaches to interpret raw data. While more precise deep learning models like YOLO and DeepSORT are available, they are computationally more expensive, limiting can limit real-time applicability. In comparison, traditional methods like HOG+SVM and Kalman Filter are computationally lighter but can break down under in complex visual situations. The simulation, even with potential drawbacks, allows for a direct comparison between detection and tracking algorithms and gives a hint about their potential strengths and weaknesses in different conditions. The following section presents the results of our experiments and compares the outcome with the potential limitations outlined in this section.

5 Results

Revisiting the objectives of this research outlines success will be gauges by three basic performance metrics, which include detection accuracy, tracking stability, and computational efficiency. These metrics were collected under the varying motion scenarios discussed earlier in section 4.4 which includes a basic linear motion path, unpredictable zig-zag motion, and occlusion events causing the objects to be temporarily blocked. By comparing the performance under these conditions we aim to quantify the strengths and weaknesses of the noted deep learning-based methods versus traditional methods.

5.1 Expected Trends and Performance Results

5.1.1 YOLO vs. HOG+SVM

As per existing research by Kaplan & Şaykol (2018) [18], YOLO is expected to outperform HOG+SVM when it comes to detection accuracy. This is an expected trend based on how each method processes visual information. Since YOLO uses a deep neural network that allows it to learn a hierarchical representation of features, it can better allow it to recognize objects in various situations, including size, orientation, and complicated background. HOG+SVM, on the other hand, relies on pre-computed edge-based features extracted via the Histogram of Oriented Gradients method, then is followed by a Support Vector Machine classification. While effective in simple cases, HOG+SVM could struggle with changes in an object's appearance and background clutter.

However, a surprising finding of Kaplan & Şaykol (2018) [18] is that YOLO also ran faster during processing when it came to FPS compared to HOG+SVM. Contrary to normal expectations that a traditional method would be less computationally complex, this suggests that optimally tuned models can be superior to more traditional methods depending on the scenario and resources available.

5.1.2 DeepSORT vs. Kalman Filter

Comparing existing research to determine how DeepSORT might compare to the Kalman Filter in accuracy can be inferred by DeepSORTs strong pairing with YOLO. One research that used YOLO and DeepSORT to detect wrong-way vehicles managed to have near 100% accuracy [19]. Using this common pairing of DeepSORT and YOLO leads to the belief that DeepSORT will outperform the Kalman Filter in tracking stability.

Previously, with YOLO and HOG+SVM, the paper noted that the deep learning model was faster than the traditional approach. It is believed that the outcome might not happen in this case with DeepSORT and the Kalman Filter as a strong reason for HOG+SVM being slower was due to a sliding window for detection. The Kalman Filter is more simple operation that should not require a lot of computational cost and prove itself to be a strong candidate in real-time performance. While DeepSORT is a deep neural network built on top of SORT and the Kalman Filter itself, which will draw drawbacks in performance speed.

5.2 Presentation of Results

Results will be presented quantitatively using:

- **Tables** showing detection accuracy and tracking ID-switch counts.
- **Bar charts or line graphs** illustrating FPS under different scenarios.
- **Trajectory or bounding-box visualizations** demonstrating how each method handles occlusions and erratic motion.

6 Conclusion

This final research project aims to compare different object detection and tracking-based approaches, including deep learning, against more traditional approaches in a controlled simulated environment through Microsoft AirSim. By comparing the object detection YOLO and HOG+SVM with the object tracking approaches DeepSORT and Kalman Filter, we aim to quantify the detection accuracy, tracking stability, and computational expense trade-offs of these approaches. Our experimental environment further includes diverse motion situations and occlusion test scenarios to facilitate a structured testing environment for the comparison of each approach under practical real-world simulating conditions.

REFERENCES

[1] N. O. Mahony, S. Campbell, A. Carvalho, S. Harapanahalli, G. Velasco-Hernandez, L. Krpalkova, D. Riordan, and J. Walsh, "Deep learning vs. traditional computer vision," in *Advances in Computer Vision: Proceedings of the 2019 Computer Vision Conference (CVC), Volume 1*. Springer Nature Switzerland AG, 2020, pp. 128–144. [Online]. Available: https://doi.org/10.48550/arXiv.1910.13796

[2] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," 2017. [Online]. Available: https://arxiv.org/abs/1705.05065

[3] O. Zendel, K. Honauer, M. Murschitz, M. Humenberger, and G. Fernandez Dominguez, "Analyzing computer vision data - the good, the bad and the ugly," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[4] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, 2005, pp. 886–893 vol. 1.

[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," 2016. [Online]. Available: https://arxiv.org/abs/1506.02640

[6] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," 2016. [Online]. Available: https://arxiv.org/abs/1612.08242

[7] ——, "Yolov3: An incremental improvement," 2018. [Online]. Available: https://arxiv.org/abs/1804.02767

[8] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[9] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," 2017. [Online]. Available: https://arxiv.org/abs/1703.07402

[10] D. A. Balamurugan, T. T. Khoei, and A. Singh, "An efficient deep learning-based model for detecting and tracking multiple objects," in *2024 2nd International Conference on Artificial Intelligence, Blockchain, and Internet of Things (AIBThings)*, 2024, pp. 1–6.

[11] P. Benoit, Y. Xing, and A. Tsourdos, "Eyes-out airborne object detector for pilots situational awareness," in *2024 IEEE Aerospace Conference*, 2024, pp. 1–8.

[12] B.-Y. Lin, C.-S. Huang, J.-M. Lin, P.-H. Liu, and K.-T. Lai, "Traffic object detection in virtual environments," in *2023 International Conference on Consumer Electronics - Taiwan (ICCE-Taiwan)*, 2023, pp. 245–246.

[13] I. Rasmussen, S. Kvalsvik, P.-A. Andersen, T. N. Aune, and D. Hagen, "Development of a novel object detection system based on synthetic data generated from unreal game engine," *Applied Sciences*, vol. 12, no. 17, 2022, visited on 2025-03-15. [Online]. Available: https://www.mdpi.com/2076-3417/12/17/8534

[14] K. Farkhodov, S.-H. Lee, J. Platos, and K.-R. Kwon, "Deep reinforcement learning tf-agent-based object tracking with virtual autonomous drone in a game engine," *IEEE Access*, vol. 11, pp. 124 129–124 138, 2023, visited on 2025-03-15.

[15] M. Wasala and T. Kryjak, "Real-time hog+svm based object detection using soc fpga for a uhd video stream," in *2022 11th Mediterranean Conference on Embedded Computing (MECO)*, 2022, pp. 1–6.

[16] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," 2017. [Online]. Available: https://arxiv.org/abs/1703.07402

[17] P. R. Gunjal, B. R. Gunjal, H. A. Shinde, S. M. Vanam, and S. S. Aher, "Moving object tracking using kalman filter," in *2018 International Conference On Advances in Communication and Computing Technology (ICACCT)*, 2018, pp. 544–547.

[18] Ö. Kaplan and E. Saykol, "Comparison of support vector machines and deep learning for vehicle detection." in *RTA-CSIT*, 2018, pp. 64–69.

[19] A. B. Siddique Mahi, F. S. Eshita, and T. Helaly, "An automated system for wrong-way vehicle detection using yolo and deepsort," in *2023 5th International Conference on Sustainable Technologies for Industry 5.0 (STI)*, 2023, pp. 1–6.

Appendix A Source Code

The following Python script demonstrates how images are captured from the AirSim environment using the drone's front-facing camera.

Listing 1: Capturing images from AirSim using Python and OpenCV

```
import airsim
import cv2
import numpy as np

# Connect to AirSim
client = airsim.MultirotorClient()
client.confirmConnection()

# Capture an RGB image from the front camera (ID 0)
responses = client.simGetImages([
   airsim.ImageRequest("0", airsim.ImageType.Scene, False, False) # Regular
      RGB image
])

# Process the image
response = responses[0]
img1d = np.frombuffer(response.image_data_uint8, dtype=np.uint8) # Convert raw
    data to array

# Ensure image is not empty
if img1d.size == 0:
   print("Error: No image data received!")
else:
   img2d = img1d.reshape(response.height, response.width, 3) # Reshape to 2D
      image

   # Convert BGR to RGB (AirSim images are in BGR format)
   img_rgb = cv2.cvtColor(img2d, cv2.COLOR_BGR2RGB)

   # Show the image in OpenCV
   cv2.imshow("RGB Camera View", img_rgb)
   cv2.waitKey(0)
   cv2.destroyAllWindows()
```

Appendix B Additional Resources

The following resources provide further information on the tools and frameworks used in this project:

- **Microsoft AirSim Documentation**: Official guide for understanding and using Microsoft AirSim for drone and autonomous vehicle simulation. Available at: `https://microsoft.github.io/AirSim/`

- **Unreal Engine Tutorials and Samples**: Epic Games' official documentation for working with Unreal Engine, which is used as the foundation for Microsoft AirSim. Available at: https://dev.epicgames.com/documentation/en-us/unreal-engine/samples-and