Hacettepe University

Computer Engineering Department

BBM 479/480 End of Project Report

Project Details

Title	Active Object Tracking on a Drone
Supervisor	Dr. Özgür Erkent

Group Members

	Full Name	**Student ID**
1	Esad Boran	21827206
2	Yusuf Koca	2200356013
3	Mustafa Emir Peker	2200356011

Table of Contents

Abstract of the Project

As drone technology continues to gain traction across multiple industries, the necessity for robust object tracking and detection capabilities becomes increasingly apparent. Traditional methods often prove inadequate, particularly when faced with adverse weather conditions or complex environments filled with obstacles. These limitations constrain the potential applications of drones. Hence, this study, titled "Active Object Tracking on a Drone," explores innovative strategies to elevate drone-based object tracking to new heights.

Drawing upon previous research, our project tackles real-world challenges such as trajectory tracking, multi-object detection, and maintaining tracking continuity despite obstacles like bridges, trees, or traffic signs. We harness the power of the Airsim simulator program, providing a virtual environment for meticulous testing and development.

Our methodology involves generating bespoke datasets tailored to our specific objectives and strategically utilizing pre-trained models to streamline the training and evaluation processes of our tracking algorithms. By immersing ourselves in diverse simulated scenarios and environmental conditions, our goal is to bolster the performance of our algorithms across a spectrum of real-world settings.

Our exploration within the Airsim simulator framework encompasses various environments and conditions. From scenarios with varying lighting conditions to simulations of obstructed vision caused by obstacles, we confront the complexities inherent in real-world drone operations, facilitating the efficient development and rigorous testing of sophisticated tracking algorithms.

We aim to demonstrate the efficacy of simulation-based approaches in addressing tangible challenges in drone-based object tracking. By enhancing reliability and adaptability, we empower drones to autonomously navigate and track targets, a capability crucial for operations such as search and rescue missions. Additionally, our work opens avenues for innovative applications, from law enforcement surveillance in urban environments to crowd monitoring during events, ultimately augmenting public safety.

In the future stages of the project, after completing development and testing within the Airsim simulator, the focus may shift to real-world implementation and testing. This phase will involve deploying the algorithms on actual drones and conducting tests in real-world conditions. By doing so, we aim to validate the effectiveness of our approaches in enhancing drone autonomy and object tracking capabilities. Through this iterative process of refinement and validation, we anticipate significant advancements in the reliability and adaptability of drone-based tracking systems.

Introduction, Problem Definition & Literature Review

Introduction

The rapid growth of drone technology has created new opportunities for research and practical applications, particularly in the realm of autonomous systems. One of the critical capabilities that enhance the functionality of drones in these applications is their ability to autonomously detect and track objects. Object detection involves identifying and locating objects within a frame, while object tracking involves following the identified object across successive frames. Integrating these capabilities into drones enhances their autonomous navigation, allowing them to interact more intelligently with their environment. This project focused on developing an active object tracking and detection system for drones using the Airsim simulator.

Problem Definition

Our first goal in this project is to create an environment asset on the Airsim Simulator. Initially, we placed multiple vehicles and assets in the environment. After setting up the environment , we implemented multiple vehicle detection and multi object tracking using artificial intelligence models. We implemented a physical tracking algorithm for the drone to follow one selected object acquired from multi object tracking. We also tried to implement a system to seamlessly switch between objects that are followed within the view of the camera. As the project progressed, we accounted for environmental variability, occlusion, and sensor clogging.

Several challenges emerged during the development process. We also faced difficulties with maintaining an appropriate distance between the drone and the tracked vehicle, ensuring the drone made precise turns, and having the drone maintain a consistent altitude during tracking. We also had problems maintaining the same Id for vehicles when there is an occlusion.

Addressing these and similar challenges required optimizing our tracking algorithms for accuracy and speed, adapting to diverse environmental conditions, and ensuring efficient use of drone resources. By overcoming these barriers , we aimed to enhance the active object tracking and detection system, contributing to the field of autonomous systems and expanding the potential applications of drones and UAV.

Literature Review

In the literature, deep reinforcement learning models, which learn to track and move UAVs together, provide state-of-the-art results. However, training these models demands extensive data, computational power, and systems to train effectively. Currently, Airsim lacks the reinforcement learning support necessary for such models, or at least support that we could use. The methodology we used for this project encompasses Airsim Simulator, Object Detection Algorithms, Object Tracking Algorithms, Multi-Object Tracking, Simulator-Based

Development, and Robustness to Occlusion and Variability. We encountered both successes and failures in these areas. We continued our detailed research on these areas throughout the period.

Airsim

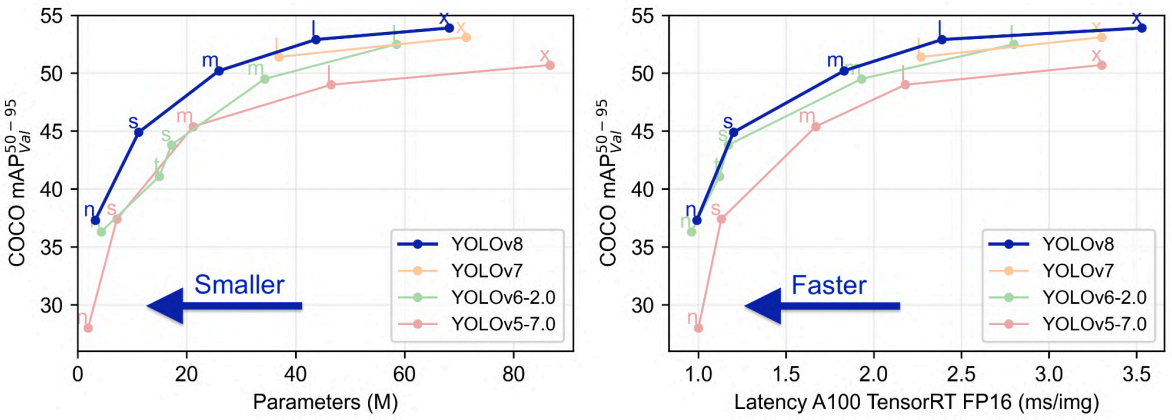
AirSim, an open-source simulator developed by Microsoft, has become a pivotal tool in the research of multi-vehicle object tracking using drones. Leveraging its high-fidelity simulation capabilities, AirSim provides a realistic virtual environment where researchers can test and refine their algorithms for tracking multiple moving vehicles. This simulator supports advanced features such as realistic vehicle dynamics, sensor models, and environmental interactions, which are crucial for developing robust tracking systems. In the context of drone applications, AirSim allows for the simulation of complex scenarios, including varied terrains, weather conditions, and urban settings, where drones must autonomously navigate and maintain visual contact with multiple targets. By offering a safe and controlled environment for experimentation, AirSim accelerates the development and testing of multi-vehicle tracking technologies, ultimately contributing to advancements in autonomous navigation, surveillance, and traffic monitoring systems.

Object Detection Algorithms

YOLOv8

YOLOv8, one of the most recent versions in the YOLO (You Only Look Once) family of object detection frameworks, expands on the unique ideas established by its predecessors. YOLOv8, like previous versions, addresses object detection as a single regression issue, predicting bounding boxes and class probabilities from entire images in a single evaluation. This approach preserves YOLO's simplicity and quickness, but YOLOv8 adds significant enhancements to improve accuracy and efficiency. It incorporates significant architectural enhancements, such as improved feature extraction and updated anchor-free detection methods, resulting in more precise object localization and categorization. YOLOv8 achieves real-time processing speeds with outstanding precision, making it appropriate for a wide range of applications, including autonomous driving and video surveillance.

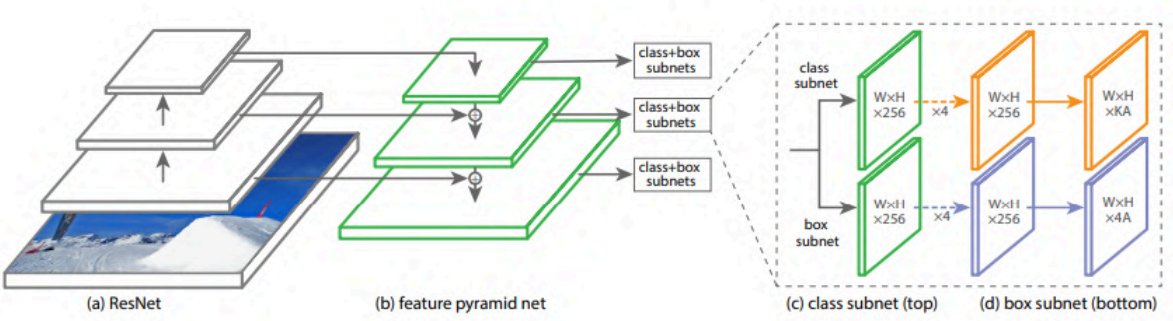
One of the most notable improvements made to YOLOv8 is its ability to handle small objects and densely packed situations more successfully than prior versions. This is accomplished through a combination of enhanced feature pyramid networks and better handling of spatial relationships within images. Furthermore, YOLOv8 employs more sophisticated training procedures and loss functions, such as advanced augmentation strategies and self-distillation, to improve model robustness and generalizability. YOLOv8 now outperforms advanced object detectors in terms of speed and accuracy, making it a top tool for real-time object detection [1].



RetinaNet

Tsung-Yi Lin et al. introduced RetinaNet, which addresses a critical object detection obstacle known as the class imbalance problem. Traditional one-stage detectors, which process a dense grid of probable object locations, frequently exhibit low accuracy because to the large quantity of background samples compared to object samples. RetinaNet addresses this issue with its new focal loss function, which decreases the loss contribution from simple, well-classified examples while focusing training on difficult, misclassified examples. RetinaNet combines the accuracy of two-stage detectors like Faster R-CNN with the efficiency of single-stage detectors, resulting in great detection performance without sacrificing speed.

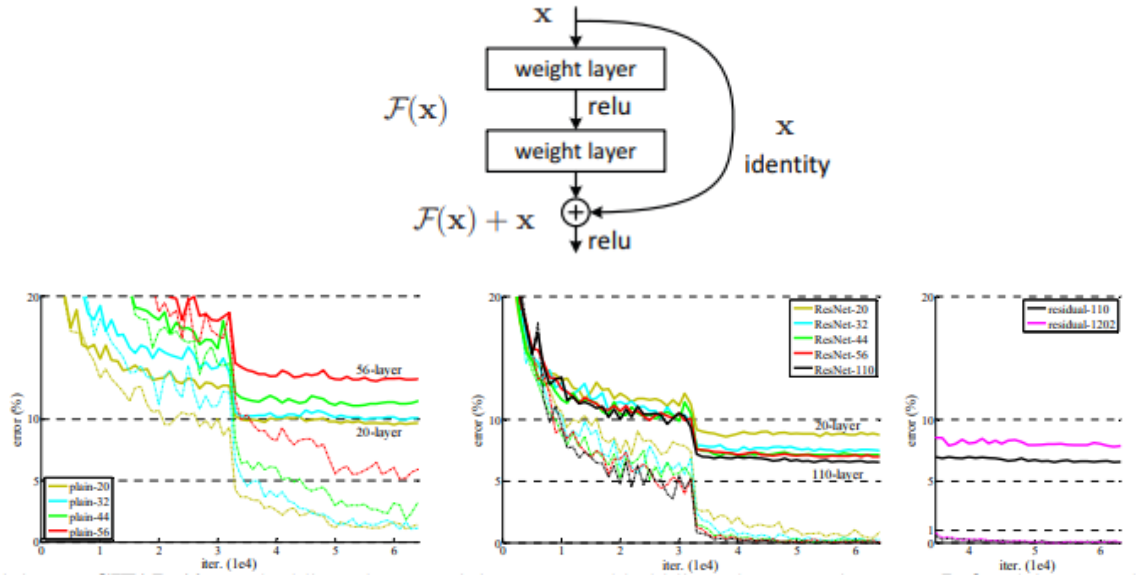
To handle objects of varying sizes, RetinaNet uses a Feature Pyramid Network (FPN) as its backbone, which is built on top of a ResNet architecture. This combination allows the backbone deep neural network such as RetinaNet to maintain great accuracy over a wide variety of object sizes. The network connects two subnetworks to the FPN: one for classifying anchor boxes and the other for regressing from anchor boxes to ground truth object boxes. This architecture enables RetinaNet to perform precise object localization and classification efficiently. RetinaNet is a premier framework for real-time object detection, producing cutting-edge performance on benchmarks like COCO. Its novel focal loss function and powerful FPN backbone contribute to this success [2].



ResNet

The Deep Residual Learning framework, presented by Kaiming He et al., solves the degradation problem faced while training very deep neural networks. The methodology allows for the training of significantly deeper networks without sacrificing performance by providing residual connections, which allow gradients to flow through the network more efficiently. Residual connections are implemented through residual blocks, adding a layer's input to its output before applying the activation function. This allows the network to learn residual functions rather than unreferenced mappings.

On benchmarks like ImageNet, residual networks (ResNets) outperformed standard convolutional networks. Beyond image classification, ResNets have been effectively used to tasks such as object detection, semantic segmentation, and image generation, establishing themselves as a core architecture in deep learning. The addition of residual connections has encouraged further developments in network designs, leading to more sophisticated models based on the concepts of residual learning [3].

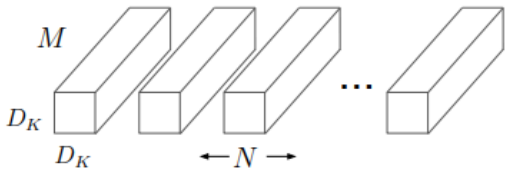


MobileNet

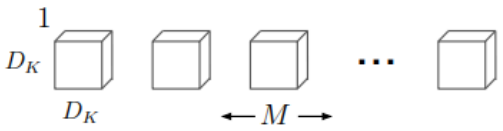
Andrew G. Howard et al. introduced MobileNets, which are efficient convolutional neural networks intended specifically for mobile and embedded vision applications. MobileNets use depth-wise separable convolutions to drastically reduce computational cost and model size, allowing real-time object detection and other vision tasks on resource-constrained devices. MobileNets' flexibility is improved by two hyperparameters: width multiplier and resolution multiplier, which balance model size, latency, and accuracy, making it suitable for varied applications.

Experiments demonstrate MobileNets' effectiveness across tasks such as ImageNet classification and object detection, maintaining high performance despite their compact size. The success of MobileNets in real-world applications, such as fine-grain classification and face attribute recognition, highlights their suitability for deployment in mobile and embedded systems. Their

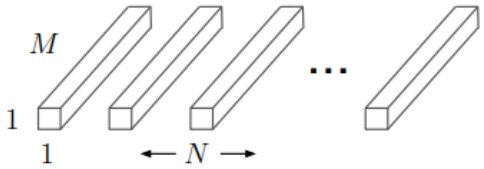
introduction has influenced both academic research and practical implementations, setting a new benchmark for efficient neural network architectures in computer vision [4].



(a) Standard Convolution Filters



(b) Depthwise Convolutional Filters



Object Tracking Algorithms

DeepSORT

DeepSORT enhances the fundamental SORT architecture by using deep appearance features for data association. This addition dramatically enhances DeepSORT's ability to handle occlusions and keep track of identities. DeepSORT uses a pre-trained convolutional neural network (CNN) for person re-identification, allowing for more accurate detecting association across frames. This decreases identity transitions while improving overall tracking performance. DeepSORT's integration of deep appearance metrics with the standard Kalman filter and Hungarian technique leads to efficient and precise tracking, making it a formidable competitor in real-time applications that require high tracking precision.

While DeepSORT improves tracking robustness, it increases computing needs due to the extra appearance feature extraction phase. This can result in slower processing speeds than simpler tracking frameworks, especially when dealing with huge amounts of video data. Using pre-trained models for appearance features may limit the tracker's capacity to adapt to new surroundings or object kinds not covered in training data [5].

		MOTA ↑	MOTP ↑	MT ↑	ML ↓	ID ↓	FM ↓	FP ↓	FN ↓	Runtime ↑
KDNT [16]*	BATCH	68.2	79.4	41.0%	19.0%	933	1093	11479	45605	0.7 Hz
LMP_p [17]*	BATCH	**71.0**	**80.2**	**46.9%**	21.9%	434	**587**	7880	**44564**	0.5 Hz
MCMOT_HDM [18]	BATCH	62.4	78.3	31.5%	24.2%	1394	1318	9855	57257	35 Hz
NOMTwSDP16 [19]	BATCH	62.2	79.6	32.5%	31.1%	**406**	642	**5119**	63352	3 Hz
EAMTT [20]	**ONLINE**	52.5	78.8	19.0%	34.9%	910	**1321**	**4407**	81223	12 Hz
POI [16]*	**ONLINE**	**66.1**	79.5	**34.0%**	20.8%	805	3093	5061	**55914**	10 Hz
SORT [12]*	**ONLINE**	59.8	**79.6**	25.4%	22.7%	1423	1835	8698	63245	**60 Hz**
Deep SORT (Ours)*	**ONLINE**	61.4	79.1	32.8%	**18.2%**	**781**	2008	12852	56668	40 Hz

ByteTrack

ByteTrack improves the tracking-by-detection paradigm by concentrating on associating low-confidence detection boxes. This method considerably enhances the recovery of tracklets that would otherwise be lost, which improves tracking accuracy. ByteTrack uses a simple Kalman filter for motion prediction and a strong association method to properly handle long-distance relationships. By combining high- and low-confidence detections, ByteTrack reduces identity switches while maintaining higher IDF1 scores, indicating its adaptability and reliability. ByteTrack excels in many tracking contexts such as MOTA, IDF1, with fewer identity shifts.

Although ByteTrack's strategy of using all available detection data enhances accuracy, it could result in additional computing expense. The inclusion of low-confidence detections complicates the association process, potentially reducing tracking performance in real-time applications. ByteTrack's efficiency depends on the quality of the underlying object detector, which may limit its performance in cases with low detection quality or changing confidence levels [6].

		MOT17			BDD100K			
Method	w/ Re-ID	MOTA↑	IDF1↑	IDs↓	mMOTA↑	mIDF1↑	IDs↓	FPS
SORT		74.6	76.9	291	30.9	41.3	10067	**30.1**
DeepSORT	✓	75.4	77.2	239	24.5	38.2	10720	13.5
MOTDT	✓	75.8	77.6	273	26.7	39.8	14520	11.1
BYTE (ours)		**76.6**	79.3	**159**	39.4	48.9	27902	29.6
BYTE (ours)	✓	76.3	**80.5**	216	**45.5**	**54.8**	**9140**	11.8

BoTSORT

BoT-SORT adds major improvements to standard tracking-by-detection algorithms by incorporating motion and appearance information, as well as camera motion adjustment. This tracker solves significant problems in SORT-like algorithms, particularly in dynamic situations with dense object tracking. BoT-SORT uses an updated Kalman filter state vector to increase the accuracy of bounding box predictions and associations. BoT-SORT improves data association by combining IoU with ReID's cosine-distance fusion, resulting in fewer identity switches and false negatives. The tracker's good results in important metrics like MOTA, IDF1, and HOTA show its robustness and reliability in retaining tracklets during occlusions and complex motion circumstances.

Despite its advantages, BoT-SORT has certain drawbacks. The computational complexity of high-density object tracking scenarios grows as advanced modules such as camera motion compensation and deep feature extraction for re-identification are integrated. This can result in

reduced processing speeds, which may not be suitable for real-time applications lacking powerful hardware. Camera motion correction relies on accurate background keypoint extraction, which can be problematic in complex scenarios [7].

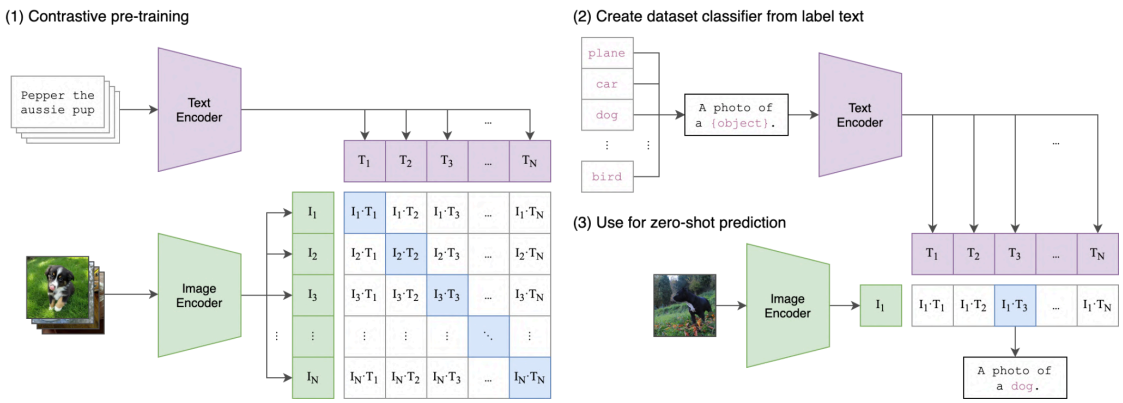
Tracker	MOTA↑	IDF1↑	HOTA↑	FP↓	FN↓	IDs↓	FPS↑
Tube_TK [30]	63.0	58.6	48.0	27060	177483	4137	3.0
MOTR [55]	65.1	66.4	-	45486	149307	2049	-
CTracker [32]	66.6	57.4	49.0	22284	160491	5529	6.8
CenterTrack [62]	67.8	64.7	52.2	18498	160332	3039	17.5
QuasiDense [31]	68.7	66.3	53.9	26589	146643	3378	20.3
TraDes [49]	69.1	63.9	52.7	20892	150060	3555	17.5
MAT [18]	69.5	63.1	53.8	30660	138741	2844	9.0
SOTMOT [60]	71.0	71.9	-	39537	118983	5184	16.0
TransCenter [50]	73.2	62.2	54.5	23112	123738	4614	1.0
GSDT [45]	73.2	66.5	55.2	26397	120666	3891	4.9
Semi-TCL [23]	73.3	73.2	59.8	22944	124980	2790	-
FairMOT [59]	73.7	72.3	59.3	27507	117477	3303	25.9
RelationTrack [53]	73.8	74.7	61.0	27999	118623	1374	8.5
PermaTrackPr [42]	73.8	68.9	55.5	28998	115104	3699	11.9
CSTrack [24]	74.9	72.6	59.3	23847	114303	3567	15.8
TransTrack [41]	75.2	63.5	54.1	50157	86442	3603	10.0
FUFET [37]	76.2	68.0	57.9	32796	98475	3237	6.8
SiamMOT [25]	76.3	72.3	-	-	-	-	12.8
CorrTracker [44]	76.5	73.6	60.7	29808	99510	3369	15.6
TransMOT [10]	76.7	75.1	61.7	36231	93150	2346	9.6
ReMOT [52]	77.0	72.0	59.7	33204	93612	2853	1.8
MAATrack [40]	79.4	75.9	62.0	37320	77661	1452	**189.1**
OCSORT [9]	78.0	77.5	63.2	**15129**	107055	1950	29.0
StrongSORT++ [12]	79.6	79.5	64.4	27876	86205	**1194**	7.1
ByteTrack [58]	80.3	77.3	63.1	25491	**83721**	2196	29.6
BoT-SORT (ours)	**80.6**	79.5	64.6	22524	85398	1257	6.6
BoT-SORT-ReID (ours)	80.5	**80.2**	**65.0**	22521	86037	1212	4.5

Re-identification (ReID):

OpenAI CLIP:

OpenAI's CLIP (Contrastive Language-picture Pre-training) model has shown great promise for re-identification tasks by utilizing its capacity to generate high-quality picture embeddings. CLIP, which is trained on a broad dataset of text-image pairs, produces embeddings that contain valuable semantic information. These embeddings can be compared for re-identification by calculating the cosine similarity between images. The use of cosine similarity provides an efficient and effective metric for measuring the similarity of embeddings, allowing for strong re-identification across diverse settings and viewpoints. Thanks to its strong discriminative strength and generalization capabilities this technique improves accuracy and reduces false matches in applications like human re-identification and object tracking.

The key advantage of using CLIP embeddings with cosine similarity is the model's capacity to recognize complex visual concepts and relationships, which is made possible by its enormous training data. This produces embeddings that are not only accurate but also adaptive to a wide range of contexts, eliminating the need for task-specific fine-tuning. However, one restriction is the computational expense of producing these embeddings, which can be resource-intensive. CLIP excels at capturing semantic information, but may struggle with distinguishing between similar items, which is critical for re-identification tasks [8].



SIFT:

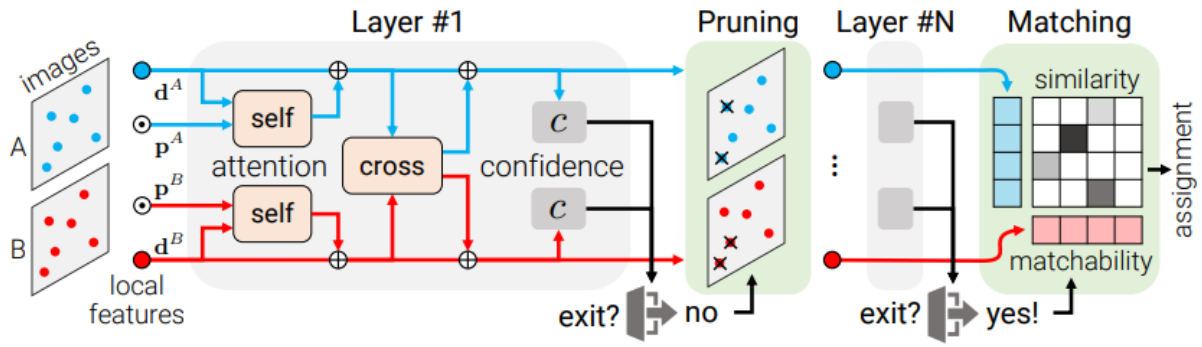
The Scale-Invariant Feature Transform (SIFT) technique is a well-known method for feature extraction and matching, particularly in re-identification applications. SIFT detects and describes local image characteristics that are scale and rotation invariant, as well as partially invariant to affine transformations and light changes. SIFT provides a reliable approach for matching features between images by extracting keypoints and computing their descriptors. This feature is especially important for re-identifying items or individuals in changing situations, such as angles, scales, or lighting.

SIFT's durability and precision make it an effective re-identification tool, allowing for precise matching even under adverse settings. However, one of SIFT's primary drawbacks is its computational complexity, which can be prohibitively expensive for real-time applications or big datasets. The algorithm's performance can suffer from major viewpoint changes or occlusions, resulting in inconsistent detection and description of local features across images.

LightGlue:

LightGlue is a contemporary feature matching technique that improves the speed and accuracy of conventional methods such as SIFT while lowering computing overhead. It uses innovations in neural network architecture to conduct efficient feature extraction and matching, making it suited for real-time applications. LightGlue improves feature matching with contextual information and learning-based techniques, resulting in more robust and trustworthy matches even in tough settings like variable lighting, scale, and viewpoint.

LightGlue's advantages include its ability to function at significantly higher speeds than traditional methods while maintaining great precision. This makes it ideal for applications that require real-time computing, such as surveillance and autonomous driving. LightGlue, on the other hand, relies on neural networks, which necessitates a large amount of training data and computer resources. The efficiency of this technique may be limited in certain re-identification settings due to variations in training data quality and diversity [9].



Kalman Filter:

The Kalman filter is a popular algorithm for determining the state of a linear dynamic system based on a set of noisy data. It is commonly used in tracking and re-identification operations due to its efficiency and simplicity. The Kalman filter works by anticipating the current state and updating it with fresh measurements, lowering uncertainty and improving the accuracy of the state estimate. This method is most successful for tracking objects or individuals with predictable motion patterns, such as pedestrians or vehicles.

One of the Kalman filter's primary advantages is its computational efficiency, which makes it ideal for real-time applications. It provides a solid framework for coping with measurement noise and successfully manages missing data. However, the Kalman filter is limited to linear systems and Gaussian noise, which may limit its use in more complex re-identification scenarios including non-linearities and non-Gaussian noise. Violations of assumptions about system dynamics and noise characteristics might negatively impact Kalman filter performance.

Extended Kalman Filter:

The Extended Kalman Filter (EKF) is an extension of the Kalman filter that is intended to handle nonlinear systems. By linearizing non-linear functions around the present estimate, the EKF can offer state estimates for systems with non-linear dynamics and observations. This makes the EKF especially effective in more complex re-identification and tracking circumstances when motion patterns and observations cannot be accurately represented by linear equations. The EKF updates the state estimate with fresh measurements, similar to the Kalman filter, but with additional steps to address non-linearities.

The EKF retains the Kalman filter's computational efficiency while expanding its application to a wider range of issues. It is commonly employed in robotics, navigation, and other applications that need precise state estimate in the presence of nonlinear dynamics. However, linearization adds complexity and possible sources of mistake. The EKF's performance is subject on linearization correctness, and it may struggle with extremely nonlinear systems or systems with significant model uncertainty. The EKF has a larger computational cost than the regular Kalman filter, potentially limiting its use in real-time applications with limited resources.

Methodology

Tools

In this project focused on drone-based vehicle detection and tracking, several key tools were utilized to achieve our objectives efficiently and effectively. These tools include Python, Unreal Engine, AirSim, and Roboflow.

1. Python

Python was the primary programming language used throughout the project due to its versatility, ease of use, and the vast array of libraries available. Its extensive ecosystem of tools and libraries made it ideal for a wide range of tasks, from data processing and analysis to the development and deployment of machine learning models.

2. Unreal Engine

Unreal Engine served as the platform for creating high-fidelity simulation environments. It is a powerful game engine that provides realistic graphics and physics, essential for developing accurate simulation scenarios.

Environment Creation: Custom environments were designed to simulate various driving conditions and scenarios, enhancing the diversity of the training data.

Scenario Control: Unreal Engine's Blueprint system allowed for the creation of interactive scenarios, which were used to test the detection and tracking algorithms under different conditions.

Unreal Engine's capabilities ensured that the simulated environments closely mimicked real-world conditions, which was crucial for training robust models.

3. AirSim

AirSim, developed by Microsoft, is a simulator for drones and other autonomous vehicles. It integrates seamlessly with Unreal Engine and provides a realistic platform for testing and data collection. It is open-source, cross-platform, and supports software-in-the-loop simulation with popular flight controllers such as PX4 & Ardupilot and hardware-in-the-loop with PX4 for physically and visually realistic simulations.

High-Fidelity Simulation: AirSim offers realistic physics and sensor models, including cameras, LiDAR, and GPS, which are critical for developing accurate detection and tracking systems.

Data Collection: Custom scripts were developed to control the drone, capture sensor data, and store it for further processing.

4. Roboflow

Roboflow is a tool for managing and preprocessing datasets, particularly useful in computer vision projects. It provides an array of features to streamline the preparation of training data.

Dataset Annotation: Roboflow facilitated the annotation of images, making it easier to create labeled datasets for training object detection models.

Data Augmentation: Various data augmentation techniques, such as rotation, flipping, and scaling, were applied to increase the diversity of the training data and improve model robustness.

Export and Integration: Roboflow allowed for easy export of datasets in formats compatible with popular frameworks such as YOLO.

Datasets

In this project, several datasets were utilized and evaluated for the task of drone-based vehicle detection and tracking. The following sections describe each dataset, their sources, and their relevance to our objectives.

1. VisDrone

VisDrone is a large-scale dataset specifically designed for visual analysis of drones. It includes annotated images and videos captured by drones, covering various scenes and scenarios.

- **Content**: The dataset contains thousands of images and videos annotated with bounding boxes for various objects, including vehicles.
- **Challenges**: Although VisDrone is comprehensive, the diversity and complexity of scenes posed significant challenges for our specific needs, particularly in terms of consistent performance in highway scenarios.

2. UAVDT

UAVDT (UAV Detection and Tracking Dataset): This dataset, available at UAVDT, provides extensive video sequences captured by UAVs, annotated for various object detection and tracking tasks.

- **Content**: UAVDT includes video sequences with annotated objects, focusing on vehicle detection and tracking in urban environments.
- **Challenges**: While the dataset is rich in urban scenarios, it lacked sufficient coverage of highway environments, which limited its applicability for our specific use case.

3. EXID (Exhaustive Highway Vehicle Dataset)

EXID: This dataset, available at EXID, focuses on vehicles on highways, providing high-quality images with detailed annotations.

- **Content**: EXID contains annotated images of vehicles on highways, making it particularly relevant for our project.
- **Challenges**: Despite its relevance, certain aspects of the dataset, such as limited diversity in vehicle types and environmental conditions, posed challenges for training robust models.

4. YOLOv8 Pre-Trained Dataset

YOLOv8 Pre-Trained Dataset: This dataset includes pre-annotated images used to train the YOLOv8 model, a state-of-the-art object detection framework.

- **Content**: The dataset comprises a wide variety of images annotated with bounding boxes for different objects, including vehicles.
- **Usage**: We utilized this dataset to fine-tune our detection models, leveraging the pre-trained weights to improve detection accuracy and reduce training time.
- **Advantages**: Using a pre-trained model significantly enhanced the performance of our vehicle detection algorithms, providing a solid foundation for further refinement.

5. Custom Dataset from AirSim Environment

Custom Dataset from AirSim Environment: We collected images from the simulated environments **created in AirSim** and used **Roboflow** to preprocess and annotate these images, creating a custom dataset.

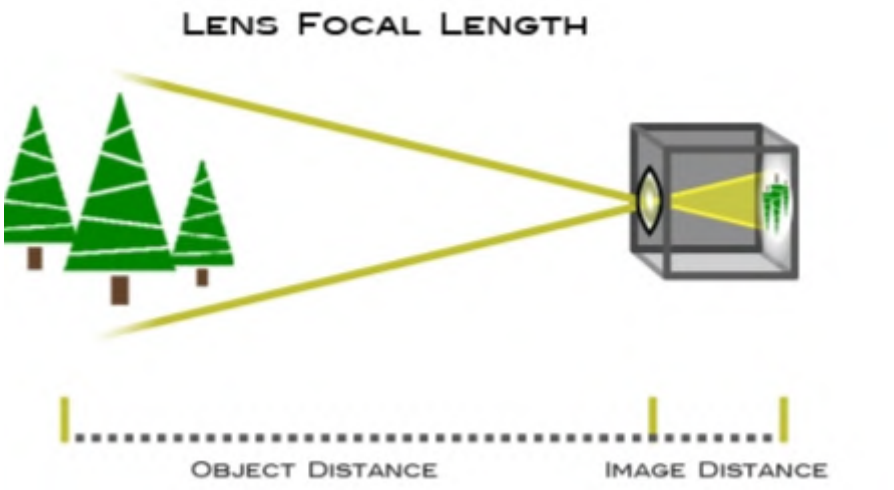
- **Content**: The dataset includes images captured in various simulated driving scenarios, annotated for vehicle detection.
- **Usage**: This custom dataset was used to supplement the YOLOv8 pre-trained model, ensuring that our models were well-adapted to the specific conditions and scenarios relevant to our project.
- **Advantages**: The ability to tailor the dataset to our exact requirements and scenarios ensured that the models were robust and performed well in the intended application environment.

While initial datasets like VisDrone, UAVDT, and EXID provided valuable insights, they were not fully utilized due to specific challenges and limitations. On top of these datasets we used our custom dataset from the AirSim environment. This combination allowed us to train highly accurate and robust vehicle detection and tracking models, tailored to the specific requirements of our project.

Detection and Tracking

Camera Calibration

In the Unreal Engine, a camera calibration environment was made. On the ground was a checkerboard. To ensure that the drone could constantly see the checkerboard, the vehicle was pushed in a square-shaped path. The camera's focal length data was extracted from the matrix, which is the calibration's output, independent of the x and y axes. The unreal engine translated the camera's pixel data to the actual distance using these focus lengths.



Object Detection

We use YOLOv8 as our main object detection method. YOLOv8 offers 60 different classes with high speed and accuracy for object detection. As most of the data used for YOLOv8 were not compatible with our case, we first fine tuned the YOLOv8-s model with datasets such as visdrone. Then upon seeing it did not perform well with the vehicles in the UE4 environment, we created our own dataset to finetune the model again. As detection is necessary in the tracking approach we took, we needed a solid detection with high accuracy. In the end we accomplished this goal.

Object Tracking

We used a tracking by detection approach. Upon research we discovered some of the popular tracking algorithms are deprecated. We decided to use BoTSORT which is well compatible with the detection architecture we chose. BoTSORT provides a robust implementation of kalman filter, camera motion compensation and a simple reid algorithm which uses some IoU operations and basic feature matching in itself. We tried different parameters to run the tracking algorithm we decided to use parameters below in our custom yaml file:

track_high_thresh: 0.6

track_low_thresh: 0.2

new_track_thresh: 0.75

track_buffer: 900

match_thresh: 0.85

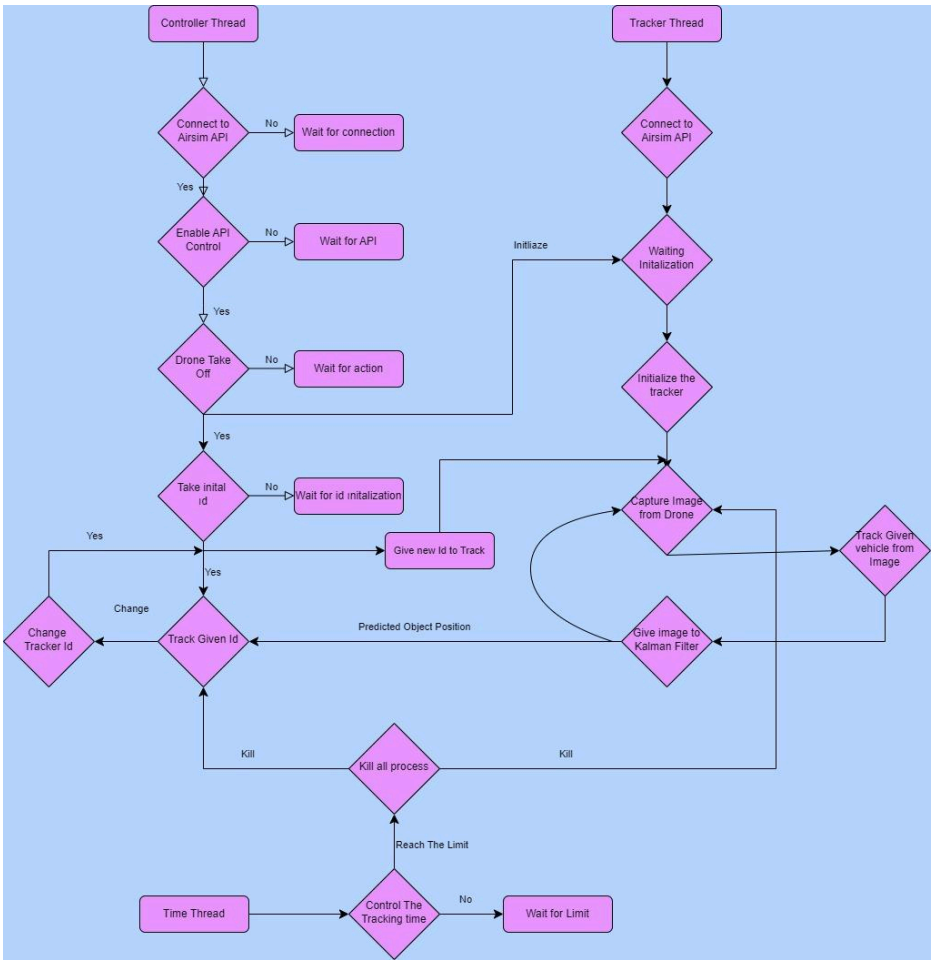
Re-Identification

After the MoT implementation was done a big challenge in our way was the occlusion problem. BoTSORT gives outstanding performance with its simple ReID implementation in clear view but we quickly realized it is ineffective when there’s a complete occlusion. In result we came up with another ReID implementation over the basic one in the tracker.

We tried several methods such as image embedding through a second deep learning model, feature matching with algorithms like SIFT, Lightglue. Some of these solutions were inefficient speedwise and some could not fix our problem as all of are objects were some sort of vehicles, each one got pretty similar scores for similarity and feature matching.

In the end we had to go back to simpler approaches and we implemented a similar approach mentioned in the paper of BoTSORT. We implemented a kalman filter for the center points of each tracked object in the screen. We calculated a car is about 40x60 pixels within the camera's view, so instead of using a xywh kalman filter we used filters for just center points and drew a 40x60 bounding box arround them. We compared IoUs to determine the reid process. We introduced a 30 frames constraint in case of no IoUs to delete unnecessary kalman filters. However since we did not implement camera movement compensation to this project yet, it still can fail on harsh movements of the camera.

Algorithm Flowchart



Results & Discussion

Tests

Drone Altitude Measurement Test

Controlling the drone and accurately determining its distance from the tracked vehicle is critically dependent on measuring the drone's altitude. We considered three options for this purpose: GPS data, LiDAR sensor, and distance sensor. Although the GPS sensor provided the highest accuracy, it was unsuitable for our specific needs, as it does not measure the relative height difference between the vehicle and the drone. Consequently, we opted for the distance sensor over the LiDAR sensor. This decision was based on our observations of a strong correlation between the drone's movements and the data obtained from the distance sensor, making it a more reliable choice for our application. Additionally, the distance sensor offered a more straightforward integration with our existing control systems, enhancing overall system performance and reliability.

Drone Control Tests

As previously mentioned, the distance between the drone and the car is calculated using camera data. The challenge in the control phase is ensuring the drone can accurately track and follow the car using this distance measurement. The formula we previously developed proved ineffective in tests, primarily because the camera's data is significantly affected by the drone's pitch movements, leading to incorrect speed estimations. To address this, we investigated the distance between the car and the drone during the object tracking phase and trained a regression model to generalize the entire drone control problem. Additionally, reinforcement learning presents a potential solution for this issue. However, despite AirSim's support for reinforcement learning, this approach remains relatively new and is too slow for effectively training a reinforcement model.

Gimbal Test

During control tests, we observed that the pitch movements of the drone adversely affected computer vision tasks. To address this issue, we integrated a gimbal into our camera system, aiming to stabilize the camera in response to the drone's movements. However, we found that the AirSim gimbal extension was not meticulously developed. The gimbal's performance is controlled by a parameter ranging from 0 to 1. Setting this parameter to 1 results in an unrealistically perfect gimbal performance. As we began to reduce the parameter to achieve a more realistic performance, we discovered that it did not function as intended. Instead of adjusting the gimbal's effectiveness, this parameter influenced the likelihood of displaying the real image versus the corrected image, leading to excessively distorted video sequences that hindered data processing.

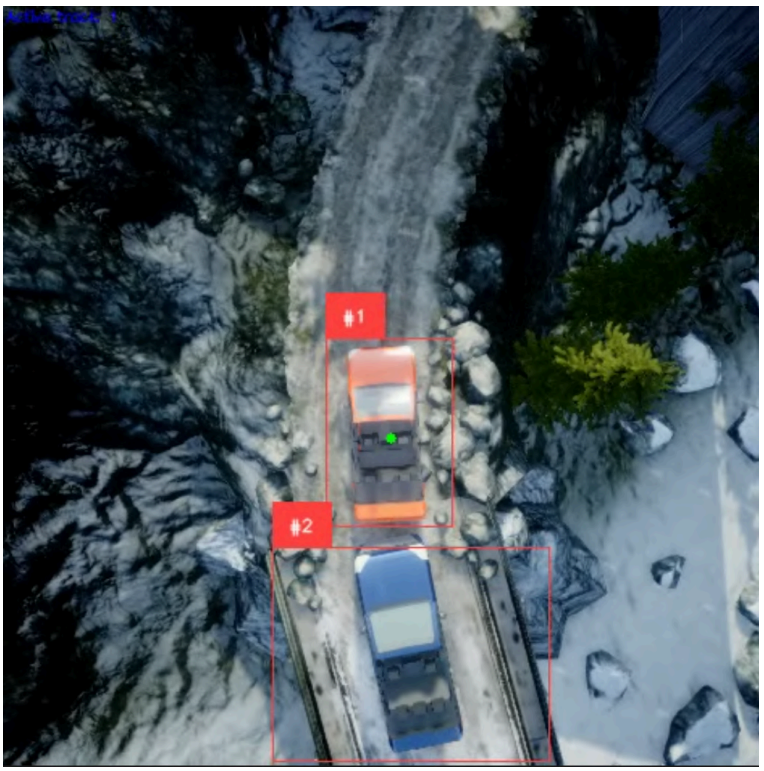
Limiting Pitch Angle

In the earlier chapters, we discussed the issues caused by the drone's very aggressive pitch movement. We therefore intended to reduce the issues we encountered by restricting the drone's pitch angle. Nevertheless, we discovered that the simulation kit we use, Airsim, lacks this feature.

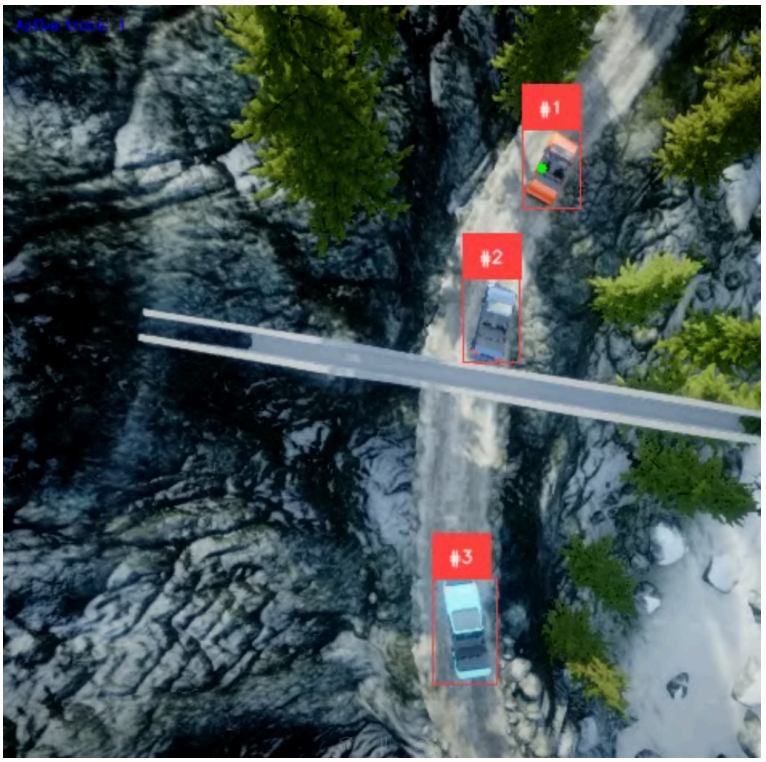
Tracker Tests

Ultimately we tested with two different trackers with different parameters. These parameters were crucial for trackers to associate tracks and match these tracks. Good set of parameters also improved re-identification task on occlusions. After obtaining near optimal parameters we tested the two trackers again.

Bytetrack gave us rather inaccurate shaped bounding boxes. This caused the tracking algorithm to fail on reid task occasionally. So we decided to not use it.

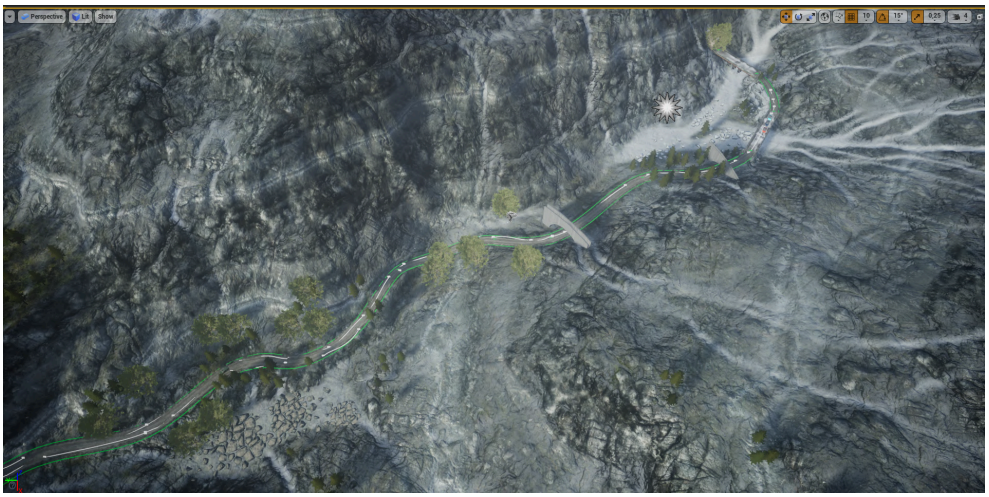


BoTSORT gave us precise results with the parameters mentioned in the methodology part. So we decided to use this tracker over bytetrack.



Results

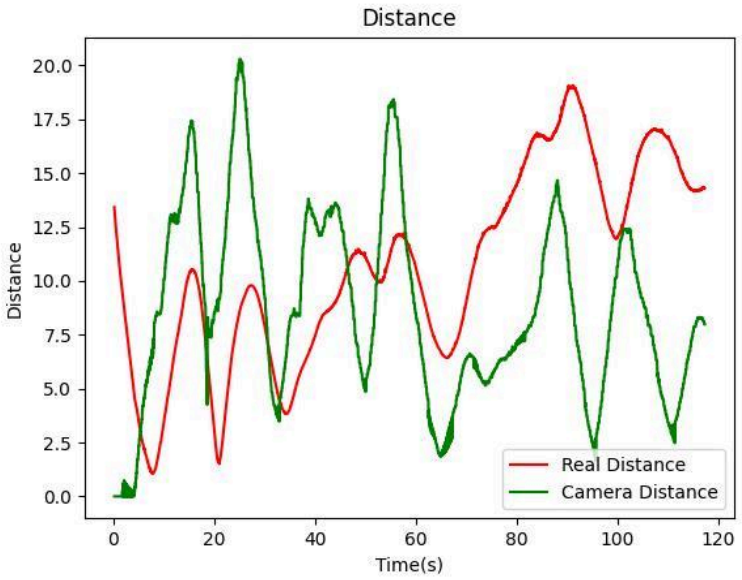
As a result of our tests, we decided on the optimal controller and tracking algorithms. We tested these choices on a route identified as the main problem. Our track starts with a slight right curve and at the apex of that curve we put a bridge to block the drone's vision. Then the road starts to slope down as a result of this the vehicles gain velocity. When the slope is finished there is a tight left turn. Our track consists of these kinds of elements such as downhill road, uphill road, trees with large leaves and traffic signs that block the drone's vision, and constantly changing reflection that occurs on the vehicle's body and on the road due to curves and slopes. As vehicles we have 4 different vehicles. 3 of them have the same body with different colors, one of them has the shape of an SUV. One of the vehicle's is coming from the opposite direction. The tracker and algorithms that we used were able to complete this route successfully. You can see our environment in the images below.



Real Distance vs Camera Distance:

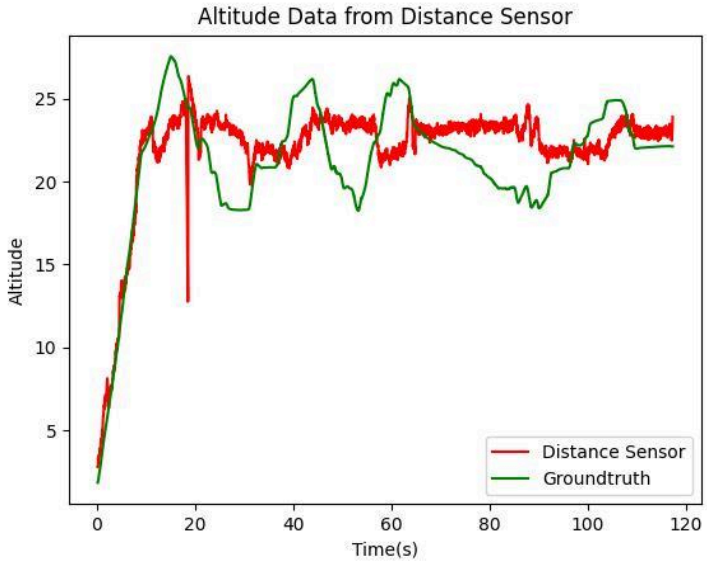
The graph of the actual distance (between the drone and the car) and the distance determined using the camera's data is displayed in the picture below.In this scenario we tracked only one car to demonstrate distance difference . Realistic values have been

computed from the camera data, as the graph illustrates. The size of the vector formed by the x and y coordinates between the drone and the car is the distance that is being discussed here.



Distance Sensor Results:

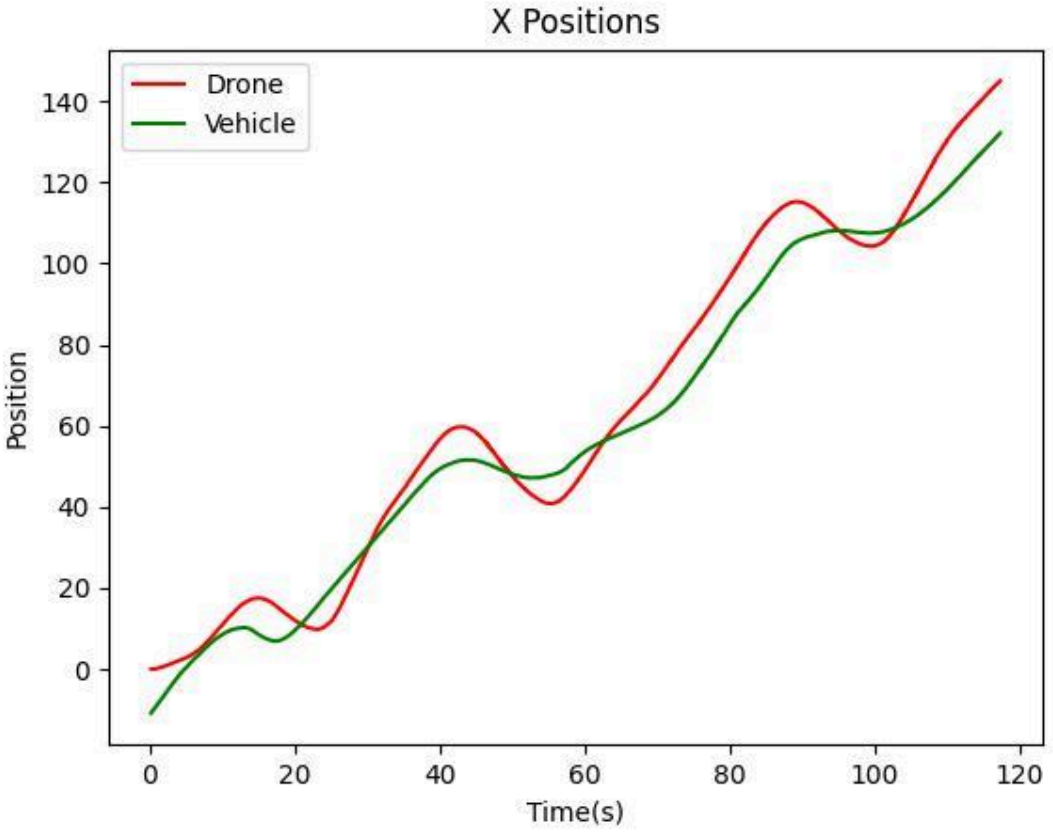
Below is the graph of the values measured by the distance sensor during this time. Since the drone was following the vehicle from behind, a gap occurred between the measured values and the actual values on the winding and inclined road.



X Position Sensor Results:

Below is the graph of the values measured by the x position sensor during this time. Since the drone was following the vehicle from behind, discrepancies occurred between the

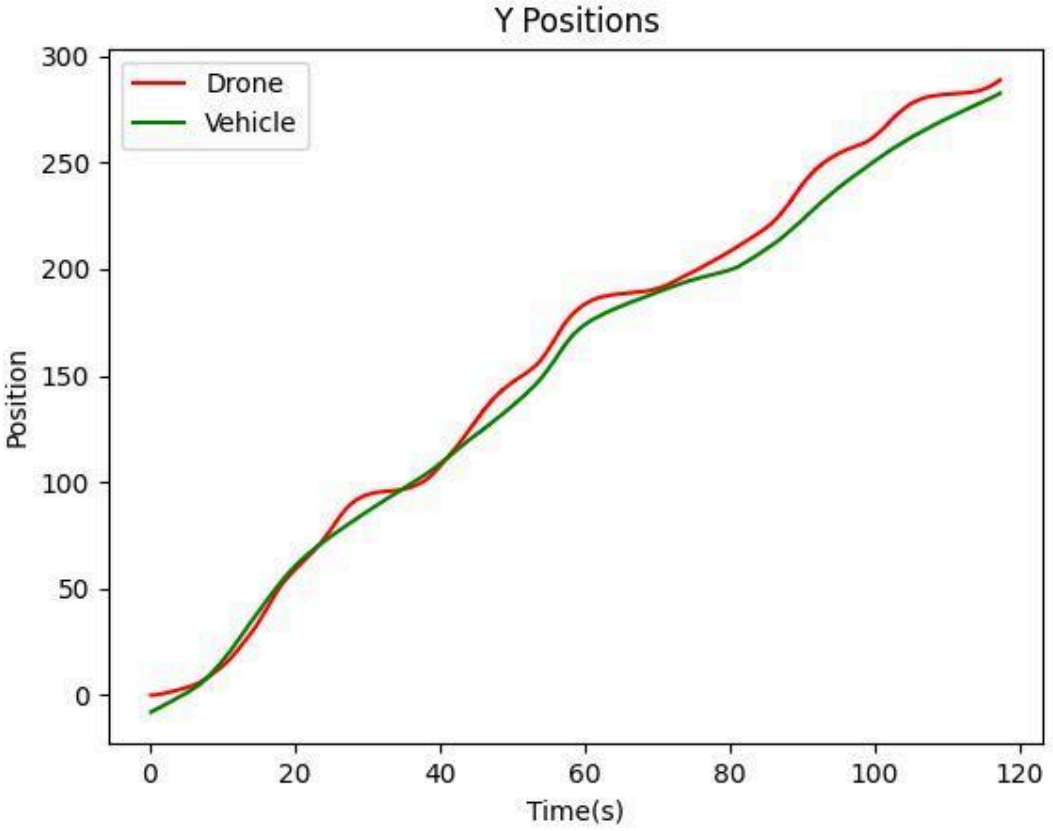
measured values and the actual values on the winding and inclined road. The actual x position, which represents the horizontal distance between the drone and the car, fluctuated significantly due to the road's curves and the car's maneuvers. The sensor data, however, captured these fluctuations with some lag and slight inaccuracies. Despite these challenges, the overall trend of the x position sensor closely followed the real-time data, demonstrating the sensor's effectiveness in tracking lateral movements .



Y Position Sensor Results:

Below is the graph of the values measured by the y position sensor during this time. The y position sensor, representing the vertical distance between the drone and the car, also showed variances from the actual values due to the road's inclines and declines. As the car navigated uphill or downhill, the y position values recorded by the sensor deviated from the true distance. These deviations were particularly noticeable on steep slopes where the drone's altitude adjustments lagged behind the car's movements. Nevertheless, the y position sensor provided a reasonable approximation of the car's vertical position, capturing the general pattern of the car's trajectory despite the inherent delays and inaccuracies.

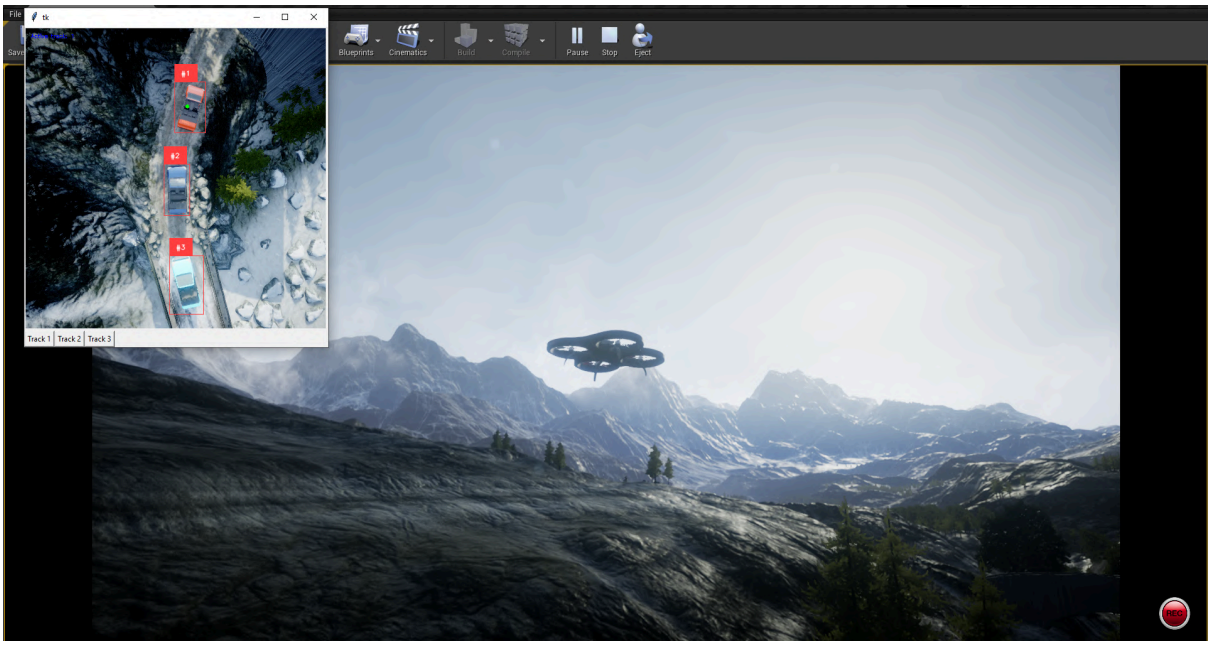
These paragraphs can be used to describe the x and y positions similar to the distance sensor results provided. If you have specific graphs or data points that need to be included, they can be referenced within these descriptions to provide more detailed insights.



DEMO

In the image below you can see that before tracking the drone takes an initial id from the user.

In the image below we can see that we are using 480x480 size images from the camera. And also we can see the buttons in the smaller window. If we click on those buttons the drone will track the vehicle that is attached to that button.



The Impact and Future Directions

Our project, which utilizes the Airsim simulator for vehicle detection and tracking, represents a significant advancement in autonomous surveillance and monitoring systems. By employing artificial intelligence (AI) to identify vehicles, our work addresses critical areas in both research and practical applications.

The AI algorithms used for vehicle detection in Airsim can be adapted to real-world conditions, offering enhanced accuracy in identifying and tracking vehicles. Transitioning from simulation to real-world application means that AI-driven drones can autonomously navigate and perform complex tasks with minimal human intervention, increasing efficiency and reducing operational costs.

One critical next step is embedding the AI algorithms into real-life drone systems. This transition will require rigorous testing and adaptation to ensure that the AI performs reliably under various environmental conditions. Research can focus on how AI algorithms react and adapt when deployed in the embedded systems of drones, accounting for factors such as weather, lighting conditions, and potential obstructions.

Currently, the drone in our project moves at a constant speed. Incorporating reinforcement learning techniques can enhance the drone's navigation capabilities. Reinforcement learning will enable the drone to learn from its environment and dynamically optimize its path, improving efficiency and adaptability in complex and changing environments.

Future research can explore both qualitative and quantitative updates to the existing system. Qualitative improvements might include enhancing the user interface for better interaction and control. Quantitative enhancements could focus on refining the AI algorithms for faster and more accurate vehicle detection. Expanding the dataset used for training the AI models can also lead to improved performance and generalizability.

In conclusion, our project has shown promising results in vehicle detection and tracking using the Airsim simulator, laying a strong foundation for real-world applications. Advancing the AI algorithms and integrating reinforcement learning will significantly enhance the functionality and applicability of autonomous drones. The future direction of this project aims at transitioning to real-life embedded systems, continuous improvement of AI capabilities, and contributing to the scientific community through publications and collaborations. These efforts will ensure that our work remains at the forefront of technological advancements in autonomous surveillance and monitoring systems.

References

[1] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2015, June 8). *You only look once: Unified, Real-Time Object Detection*.

[2] Lin, T. Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017, August 7). *Focal Loss for Dense Object Detection*.

[3] He, K., Zhang, X., Ren, S., & Sun, J. (2015, December 10). *Deep Residual Learning for Image Recognition*.

[4] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017, April 17). *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*.

[5] Wojke, N., Bewley, A., & Paulus, D. (2017, March 21). *Simple Online and Realtime Tracking with a Deep Association Metric*.

[6] Zhang, Y., Sun, P., Jiang, Y., Yu, D., Weng, F., Yuan, Z., Luo, P., Liu, W., & Wang, X. (2021, October 13). *ByteTrack: Multi-Object Tracking by Associating Every Detection Box*.

[7] Aharon, N., Orfaig, R., & Bobrovsky, B. Z. (2022, June 29). *BoT-SORT: Robust Associations Multi-Pedestrian Tracking*.

[8] Openai. (n.d.). *GitHub - openai/CLIP: CLIP (Contrastive Language-Image Pretraining), Predict the most relevant text snippet given an image*.

[9] Lindenberger, P., Sarlin, P. E., & Pollefeys, M. (2023, June 23). *LightGlue: Local Feature Matching at Light Speed*.