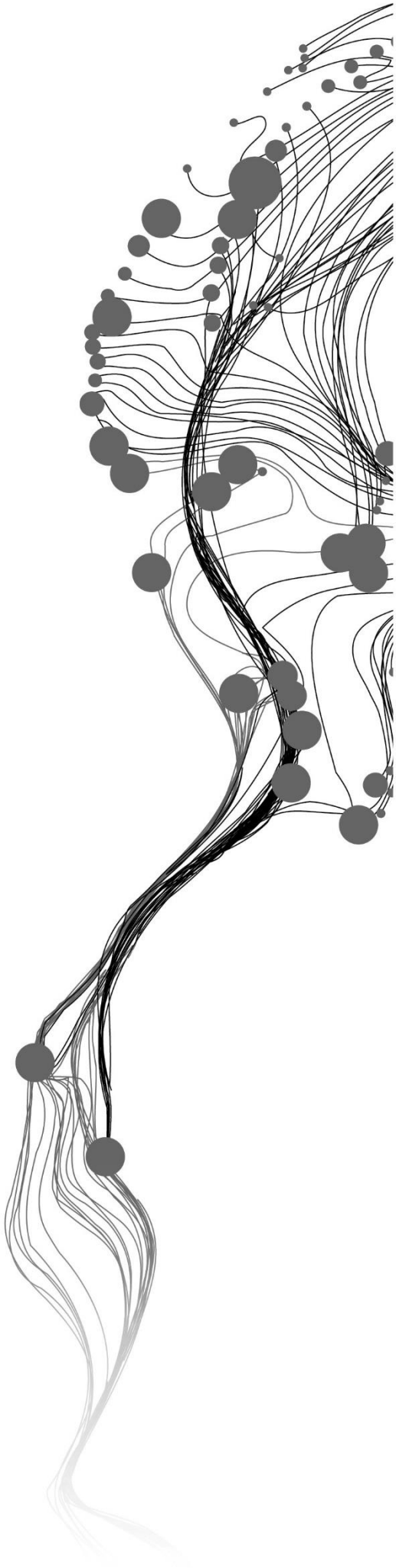
Using synthetic data to improve the performance of UAV-based vehicle detection and speed estimation models

YAOZHENG SONG
MAY, 2025

SUPERVISORS:
Prof.Dr.Ing. F.C. Nex
Dr. Sofia Tilon



Using synthetic data to improve the performance of UAV-based vehicle detection and speed estimation models

YAOZHENG SONG
Enschede, The Netherlands, May, 2025

Thesis submitted to the Faculty of Geo-Information Science and Earth Observation of the University of Twente in partial fulfilment of the requirements for the degree of Master of Science in Geo-information Science and Earth Observation.
Specialization: Geoinformatics (GFM)

SUPERVISORS:
Prof.Dr.Ing. F.C. Nex
Dr. Sofia Tilon

THESIS ASSESSMENT BOARD:
Prof.Dr.Ir. M.G. Vosselman
Dr. M.N. Koeva

ABSTRACT

UAVs can flexibly monitor roads and have a broad range of applications in the field of vehicle speed estimation and traffic monitoring. However, existing UAV datasets are very limited, especially aerial data with vehicle annotations are rare and the coverage of the scene is narrow, which hinders the improvement of the performance of data-driven deep learning models. To address these issues, a semi-automated process is proposed to reconstruct a real outdoor scene, acquire and generate synthetic training data with annotations, and use these data to train deep learning models to improve their vehicle speed estimation performance. Our process is divided into four main steps: scene reconstruction, data acquisition and annotation, model training, and testing and comparing model performance. This method uses synthetic data to overcome the limitations of real data by enriching the training samples with realistic synthetic images and highly accurate annotations. Such synthetic images have been proved that they can significantly improve the robustness and accuracy of the model. In our experiments, models trained with synthetic datasets generalise well to real UAV videos, while models pre-trained with real datasets and fine-tuned with synthetic datasets have further improved performance. The results show that synthetic data can improve the accuracy of detection and speed estimation of deep learning models.

Keywords: UAV; synthetic data; YOLOv8, vehicle detection; vehicle tracking; speed estimation

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to all those who contributed to the completion of this study.

First of all, I would like to express my heartfelt gratitude to my first supervisor, Prof.Dr.Ing. F.C. Nex, for his consistent and unwavering support and professional guidance. His forward-looking academic vision, as well as his knowledgeable, rigorous and accommodating academic wisdom, not only pointed the way for my research, but also provided me with crucial answers whenever I was confused. I am especially grateful for his extraordinary patience and invaluable advice, and his continuous encouragement was an important pillar for the completion of this study.

I sincerely thank my second supervisor, Dr. Sofia Tilon, for her meticulous guidance during the research process. Her rigorous control and guidance on experimental design, data analysis, and review of drafts helped me get through the most difficult phase of the study.

I am particularly grateful to M.C.F. Metz – Bekkers and Drs. J.P.G. Bakx, our study adviser and course director, whose constant reminders and systematic progress tracking ensured the completion of the project.

Thank you to my family and friends Jay, Enting and Xuanya, your support and companionship was a strong support during the research process.

Finally, thanks to ITC for providing a strong academic infrastructure that provided a solid foundation for the successful completion of this study.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1. Background information

Currently, Unmanned Aerial Vehicles (UAVs) are widely utilised in various fields due to their unique advantages, including flexibility, ease of deployment, and cost-effectiveness. In agriculture, they assist farmers in precisely monitoring crop growth conditions to optimise farm operations (K. Li et al., 2023) (Srivastava & Prakash, 2023). In the realm of safety and surveillance, UAVs are employed for search and rescue operations, as well as for inspecting large-scale structures like bridges for structural health monitoring(Lapointe et al., 2022). In the field of transport, compared with the relatively fixed and small observation area of traditional traffic cameras, the characteristics of drones capable of surveying large areas at low cost are more obvious(Tilon & Nex, n.d.). For instance, AI-equipped UAVs automatically detect and localise road damage through target detection and localisation in drone imagery(Silva et al., 2023). Additionally, in the field of traffic monitoring, UAVs have also shown promising applications, especially in vehicle speed detection. Acquiring vehicle speeds quickly can provide important information for traffic management in order to assist the transport sector in coping with the growing traffic problems caused by the increase in the number of vehicles. In the field of traffic monitoring, UAVs have also shown promising applications, especially in vehicle speed detection. Acquiring vehicle speeds quickly can provide important information for traffic management to assist transport authorities in coping with the increasing traffic problems caused by the increase in the number of vehicles. When combined with deep learning models, vehicle speed detection using UAVs can be divided into three sub-parts, vehicle detection, vehicle tracking and vehicle speed estimation. In order to obtain good results for vehicle speed detection, it is mandatory to train and obtain a well-performing deep learning model using sufficient data (Balamuralidhar et al., 2021).

Many of the UAV applications mentioned above rely to varying degrees on the corresponding deep learning models. In contrast, the performance of deep learning models needs to be improved by training them using large amounts of data. (P. Zhu et al., 2022). However, due to local legal restrictions or adverse weather conditions, UAVs may be unable to fly into specific areas, resulting in a shortage of adequate qualitative data for those regions (Andle et al., 2023). In the latest regulations issued by the European Union Aviation Safety Agency (EASA) in 2022 (*EUR-Lex - 02019R0947-20220404 - EN - EUR-Lex*, n.d.), EASA member states will delineate 'UAS geographical zones', which are areas established by competent authorities to restrict or exclude drone flights. In the Netherlands, flying over government buildings, military terrains, and nature reserves, among others, is strictly prohibited. For lightweight drones, their flight altitude above ground level should not exceed 30 meters. As for heavy drones, they must maintain a distance of no less than 120 meters from the Earth's surface. Within permissible flight zones, flying around railways and roads with speed limits exceeding 60 kilometres per hour is prohibited (*Waar Mag Ik Niet Vliegen Met Een Drone? | Rijksoverheid.Nl*, n.d.). Moreover, for particular points of interest (e.g., potholes, cracks, and ruts in road damage detection) (Arya et al., 2022), their occurrence in the environment is relatively scarce, leading to data imbalance for training deep learning models. To alleviate

the current shortage of datasets and to address data imbalances, synthetic datasets are increasingly seen as an acceptable alternative. Generally, synthetic data is considered to be a type of data that is generated by computer programs and algorithms to simulate the real world. When real data is lacking, simulated data can be considered as a substitute and used to train deep learning model (Y. Lu et al., 2023). However, collecting synthetic data can be a challenge in itself. At this point, reconstruction of the synthetic environment plays an important role in obtaining synthetic data. Synthetic environments can be digital replicas of the real world that reflect the changes in the real world to some extent, simulate rare or difficult-to-access scenarios in the real world at a relatively low cost, and provide some interactivity. This means that a complete synthetic scene can provide a large amount of compliant synthetic data quickly and efficiently (Nikolenko, 2019).

To address these challenges, this thesis develops a semi-automated method for reconstructing real-world outdoor scenes in a simulation platform and using these models to generate synthetic UAV images. The synthetic images are automatically annotated (e.g. vehicle bounding boxes) and used to train deep learning models for vehicle detection and tracking. By using synthetic data to train and fine-tune the deep learning models, the accuracy of vehicle detection is improved and more reliable speed estimation is achieved when applying the models to realistic data

1.2. Objectives and research questions

1.2.1. Problem statement

The data collection challenges for vehicle detection, tracking, and speed estimation in the Unmanned Aerial Vehicle (UAV) domain pose a significant obstacle to training deep learning models. In this regard, there are two key issues:

1. Lack of ground truth data for vehicle detection and vehicle speed estimation.
The performance of the deep learning model is highly dependent on the availability of high-quality ground truth data. In some areas, the necessary training data regarding vehicle detection and speed estimation are insufficient due to local regulations or weather conditions. Some datasets provide usable resources, but it is still difficult for them to cover all potential points of interest. For example, Visdrone (P. Zhu et al., 2022), a UAV dataset collected mainly in Chinese cities and villages, may make it difficult to include specific situations in other parts of the world.
2. Lack of ground truth data for vehicle detection and vehicle speed estimation.
Existing datasets generally lack sufficient samples of rare or edge scenarios, such as unusual traffic conditions, rare weather conditions, or special pavement types (Rahmani et al., 2024). Such weaknesses may limit the reliability and accuracy of the models in a variety of realistic scenarios.
3. Lack of solutions for the recreation of existing outdoor scenes in simulation software.
Accurate and realistic reconstruction of existing scenes is critical for meaningful simulations and effective algorithm training. However, current methods are not sufficiently detailed to fully replicate outdoor scenes, which leads to discrepancies between simulated environments and

reality. For instance, it may be difficult for synthetic data to fully simulate real-world lighting and reflective conditions as well as highly enriched textures, which are key factors affecting deep learning models in object detection tasks (Bai et al., 2024). Therefore, this discrepancy will potentially affect the performance of deep learning methods applied to vehicle detection and speed estimation in real-world environments (Y. Lu et al., 2023).

4. Lack of datasets for information on the dynamics of drones and vehicles.

Many existing datasets do not contain enough information about the dynamics between the UAV platform and the moving vehicle. For example, the height, attitude, and speed information of the UAV platform as well as the continuous drastic changes in the background brought about by its movement (Ibrahim & Delibaşoğlu, 2021); the changes in the image of the vehicle due to the movement of the UAV (perspective, scale) as well as the dragging and blurring of the vehicle as it moves (L. Lu & Dai, 2024). The lack of similar data in the training dataset may lead to unstable vehicle speed estimation based on UAV images and systematic errors.

1.2.2. Research Objectives

In order to address the problem of recreating existing outdoor infrastructure scenarios and training data shortage and imbalance and to improve the performance of deep learning models to detect vehicles and estimate their speed, this study consists of the following objectives：

1 To semi-automatically reconstruct a realistic outdoor environment for generating annotated synthetic data.

- What is the suitable method for the semi-automatic reconstruction of the scenarios in this study?
- How to acquire data with pre- annotated in the reconstructed scene?

2 To improve the vehicle speed estimation performance of deep learning model by training with synthetic data.

- Can deep learning models trained and fine-tuned with synthetic datasets be generalised to real-world scenarios in the vehicle speed estimation task?
- To what extent will vehicle speed estimation performance be improved after fine-tuning using synthetic data?

2. RELATED WORK

This section will present existing work in the areas of vehicle speed detection using drones, synthetic data generation methods, and the application of synthetic data in augmenting deep learning models for traffic-related tasks.

2.1. Vehicle Speed Detection

Vehicle speed detection typically involves three key steps: vehicle detection, vehicle tracking, and speed estimation.

2.1.1. Vehicle Detection

Deep learning-based vehicle detection methods for UAVs can be broadly divided into one-stage and two-stage algorithms (Carranza-García et al., 2020). Two-stage algorithms divide vehicle detection into two steps. Firstly, generating candidate regions (called region proposal). This step identifies potential regions in the image that may contain objects, thus allowing the algorithm to narrow down the search to specific regions of interest. In the second stage, these candidate regions are classified to determine whether they contain vehicles or other objects. This approach ensures higher detection accuracy as the object classification task is performed on a smaller, finer set of candidate regions. The two-stage approach typically performs well in situations where detection accuracy is critical but may be slower compared to the one-stage approach.

Two-stage algorithms are typically based on R-CNNs (Girshick et al., 2013). For example, by introducing a new anchor generation method based on vehicle speed and using Q-squared penalty coefficient optimisation, i.e., optimising the first step to improve the performance of the Faster R-CNN model when the vehicle is occluded or when the speed varies considerably (Cui et al., 2019). Another study also improved the model's vehicle detection performance by improving the first step by suggesting the entire input image and a set of objects instead of each region(Yin et al., 2022).

One-stage methods, such as YOLO (Redmon et al., 2016), This method processes the entire image at once, skipping the region proposal stage and performing object detection in a single step. One-stage methods are faster than two-stage methods because they do not need to generate region proposals before classification. However, compared to two-step methods, their accuracy may be insufficient, especially when dealing with smaller or overlapping objects (Carranza-García et al., 2020).

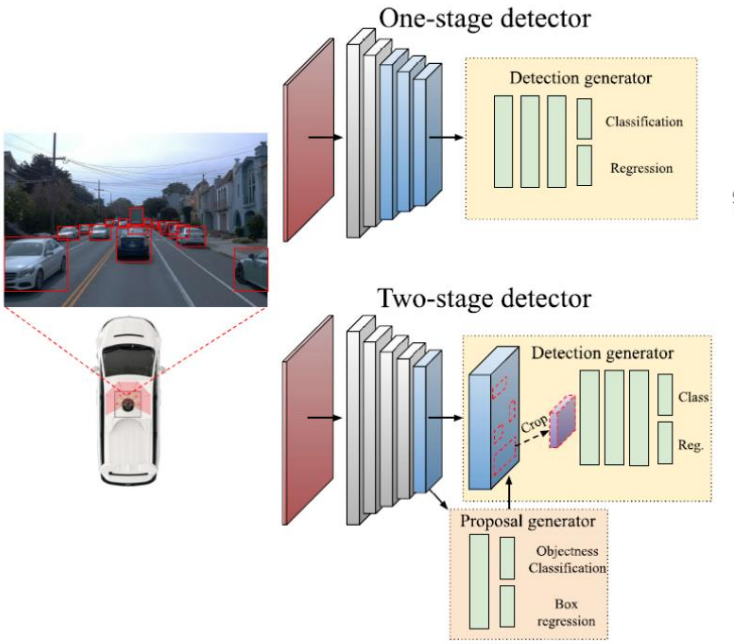


Figure 1: One-stage detection architecture of RetinaNet and Two-stage detection architecture of Faster R-CNN (Carranza-García et al., 2020)

YOLO, in particular, is widely used in vehicle detection tasks due to its speed and accuracy. A recent study applied meta-heuristic algorithms to optimise YOLO by independently optimising the hyperparameters of the YOLO model using the Grey Wolf Optimiser (GWO), the Artificial Rabbit Optimiser (ARO) and the Chimpanzee Leader's Selection Optimisation (CLEO) in order to detect the vehicle's performance under adverse weather conditions such as fog and heavy rain (Özcan et al., 2024). In addition, due to its lightweight nature, YOLO can run on edge devices, especially UAVs, and a model of YOLO specifically designed to be mounted on UAVs is already available. Aero-YOLO is a lightweight version of YOLOv8 designed for UAV-based vehicle detection. The model combines GSConv and a stochastic attention mechanism to enhance the detection and extraction of small vehicle features in aerial imagery and is evaluated on the UAV-ROD and VisDrone2019 datasets, which confirms its improved accuracy and speed of the vehicle and pedestrian detection (Shao et al., 2024). As of this writing, the latest version, YOLOv8, is currently the most advanced and can process video frames in real-time, extracting features to detect vehicles efficiently (Reis et al., 2023; Terven & Cordova-Esparaza, 2023).

2.1.2. Vehicle Tracking

The framework for vehicle tracking has been divided into two categories: non-detectable tracking and detectable tracking (W. Luo et al., 2021). Undetectable tracking requires manual setting of the initial tracking target and is currently only applicable to tracking targets that have already been specified. Most of the mainstream vehicle tracking methods are implemented based on detection tracking (B. Yang et al., 2020), which involves first detecting target table objects on frames with video decomposition, extracting features that includes target edges, shapes, colours, textures, optical flow and HOG (Histogram of Oriented Gradients) (Abdallah et al., 2022; Dhatbale et al., 2021), centre of mass or borders in the image,

and then tracking the target by matching the feature points or regions with other frames (Aishwarya & Kulkarni, 2021).

Fast R-CNN (Ren et al., 2015) and YOLO (B. Yang et al., 2020) are the most commonly used target detectors in this method, thanks to their fast detection speed, and then using the corresponding matching algorithms, such as Re-identification (RE-ID) (Lin et al., 2019) and intersection over union (IOU) (Wu et al., 2022), to match feature points from different frames to achieve the vehicle tracking process. In addition to tracking by comparing features between frames, deep learning models can be used to predict (Zarindast & Sharma, 2023) the position of the vehicle based on the detection results of the first frame using some motion regression algorithms such as the Hungarian algorithm and the Kalman filter (Tyagi et al., 2023), and then updating the position based on the detection results. This makes this approach more real-time and relatively less computationally expensive. The classical models based on this idea are DeepSORT (Wojke et al., 2018) and MOSSE (Dardagan et al., 2021).

DeepSORT (Deep Simple Online Realtime Tracking) is an extension of the SORT (Simple Online Realtime Tracking) algorithm that enhances its tracking performance by integrating deep learning capabilities into the SORT algorithm. Combining DeepSORT as a tracker with YOLOv5 (as a detector) facilitates real-time traffic management through efficient vehicle detection and vehicle tracking.

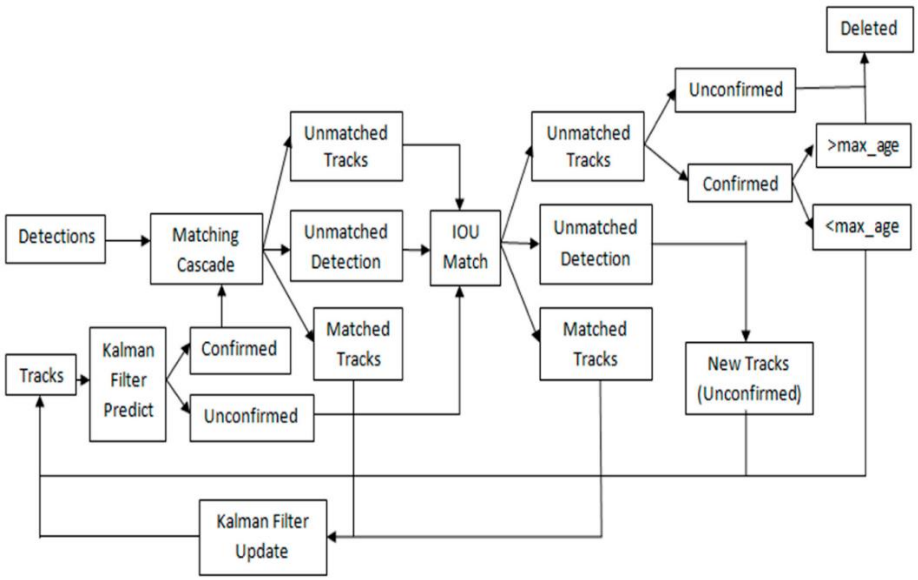


Figure 2: Vehicle tracking based on DeepSORT algorithm (K. Li et al., 2023)

The authors and others demonstrated superior tracking performance by testing DeepSORT on BDD100K and PASCAL datasets (K. Li et al., 2023). MOSSE (Minimum Output Sum of Squared Error) Target tracking is achieved by calculating the correlation through the filter, and the position of the largest response in the returned output response is the centre of the desired target (Huo et al., 2022). In another study on traffic monitoring, MOSSE, due to its nature of running on the CPU, enabled it to not interfere with target detection algorithms running on the GPU and was able to achieve good accuracy at higher execution speeds on a UAV platform (Balamuralidhar et al., 2021).

2.1.3. Speed Estimation

Vision-based vehicle speed estimation is based on vehicle detection and vehicle tracking. After the detection and tracking are done, the focus of vehicle speed estimation then shifts to obtaining the variable: metre-to-pixel ratio (scale factor), which is the distance to the part of reality corresponding to each pixel. For different camera systems (monocular and stereo camera systems), the method of obtaining the scale factor varies.(Llorca et al., 2021).

In the stereo vision system, the acquisition of scale factor relies on parallax calculation. For two cameras in the stereo vision system that capture feature points in the image of the same scene are matched, the parallax of the corresponding points in the image is calculated, then the depth information of the image is obtained based on the parallax and the baseline of the camera, and finally, the scale factor is obtained by the relationship between the depth information (the distance in the real world) and the position of the pixels in the image (Jiang et al., 2019). When the scale factor is obtained, the estimate of vehicle speed can be converted to the number of pixel points per unit time of vehicle movement in the video or image when knowing the frame rate of the video or the exact timestamp of the image(Jiang et al., 2019). The stereo vision system can directly acquire the 3D depth information of the scene, and the scale factor for velocity estimation can be directly determined by calculating the depth of the object and the pixel movement, with a higher accuracy compared to the monocular system (Llorca et al., 2021). For example, this study of speed estimation based on a stereo vision system can achieve a speed estimation error of less than 10 per cent at speeds of up to 30km/h (Jiang et al., 2019). Another study achieved a maximum error of -3.24 per cent in the estimation of single-vehicle speed at about 46km/h (L. Yang et al., 2019).

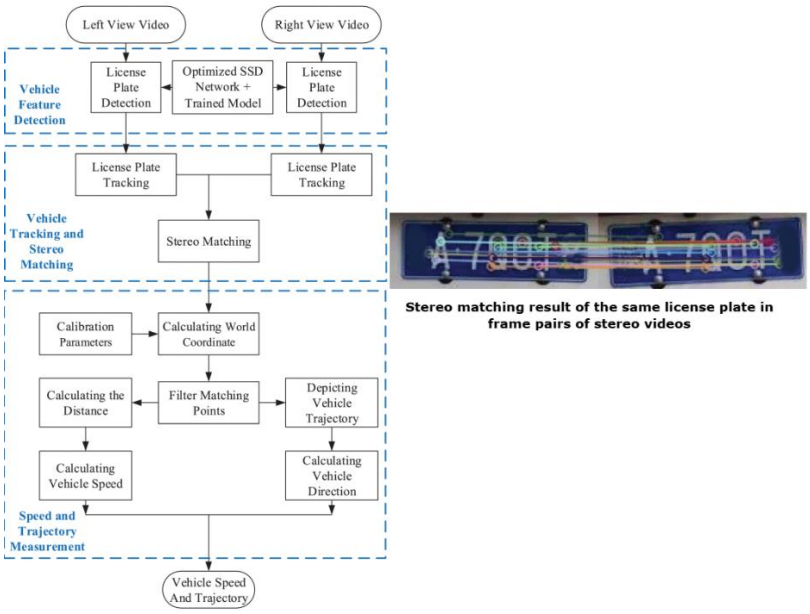


Figure 3: Example of a stereo vision system that calculates vehicle speed by detecting and matching licence plate feature points, where the left side is the vehicle speed calculation process, and the right side is a sample result of the matching of licence plate feature points (L. Yang et al., 2019).

Since there is only one camera in the monocular vision system, it is not able to derive the depth information and obtain the scale factor directly through parallax as in binocular vision. When the target scene conforms to homography properties, such as a straight and flat road surface, the scale factor can be obtained by a relatively simple linear perspective transformation (T. Huang, 2018). At this point, the estimation of speed will remain as a calculation of the number of pixel points of the detected vehicle moving between frames (Gunawan et al., 2019). It can also be used to assist in obtaining the scale factor when the size of some objects in the input video or picture is known. For instance, when the size of the licence plate is known and the frame rate of the video is constant, the distance between the licence plate and the camera can be inverted from the size of the licence plate in the image when the internal and external orientation parameters of the camera are determined (Vakili et al., 2020). Another case in point is the uniform grid lines on the road, which can be used to help generate a virtual reference plane that corresponds exactly to the grid lines for estimating vehicle speed (Kim et al., 2018). In addition, inserting lines of intrusion or regions of interest (ROIs) in the input video or image is also a common approach. For example, this study inserted a line of incursion into the target area, at which point the speed estimate would be converted from counting pixel points to the number of frames required for the target vehicle to pass through the line of incursion (Javadi et al., 2019). If multiple equidistant intrusion lines are set up connecting the two sides of the road at realistic distances, the vehicle speed problem can also be converted to the time it takes to reach the next intrusion line from one line since the distance between the intrusion lines is known (Dahl & Javadi, 2019). ROIs are used in a similar way to equidistant intrusion lines, in that the distance from the start point to the end point of the ROI is fixed, and the estimated vehicle speed can be obtained by the time the vehicle takes from the start to the end (Shaqib et al., 2024).

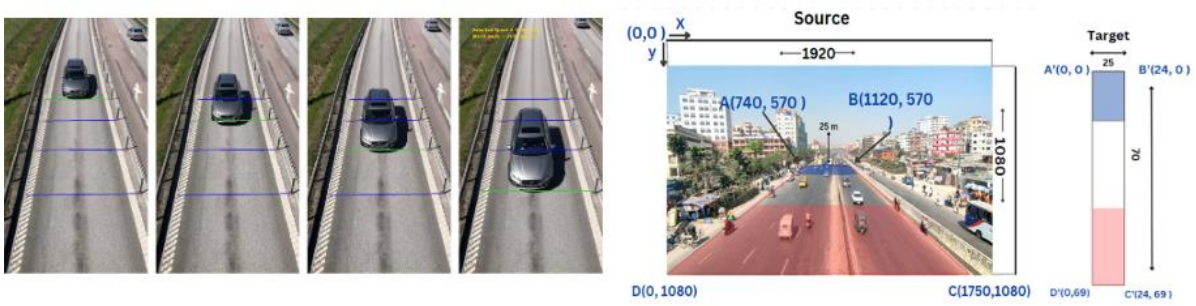


Figure 4: Example of Intrusion Lines (left) (Javadi et al., 2019) and ROI (right) (Shaqib et al., 2024)

2.2. Synthetic Data

Synthetic data has become an increasingly popular tool in the current field of deep learning (Nikolenko, 2019). Synthetic datasets are usually considered to be data that are not obtained through direct observation and are simulated by computer programs or algorithms to generate. Synthetic datasets were first used in 1989 in autonomous driving (Pomerleau, 1989) and optical flow analysis (Little & Verri, 1989) and have evolved with the development of computer vision technology. Compared with real data, synthetic data can usually provide enough data for training, and some synthetic data are capable of automatic labelling, which will greatly improve the efficiency of creating datasets and significantly reduce the cost.

2.2.1. Methods for creating synthetic data

The creation of synthetic data can be accomplished in a number of ways, each of which is appropriate for a particular type of application. One method involves the use of a scenario generator, a piece of software or tool specifically designed to create or simulate specific environments and things. It allows the user to create the desired virtual environment by specifying parameters and constraints (Musgrove et al., 2014). For example, Infinigen, a library of procedural rules and utilities built on Blender. It generates assets programmatically, enabling infinite variations and combinations using stochastic mathematical rules, and generates images using Blender rendering (Raistrick et al., 2023). Another approach is programming, where synthetic data are generated directly by algorithms, such as in early optical flow analyses (Little & Verri, 1989), programmatically generated synthetic image data that met the requirements were used to test and validate the validity of the model.

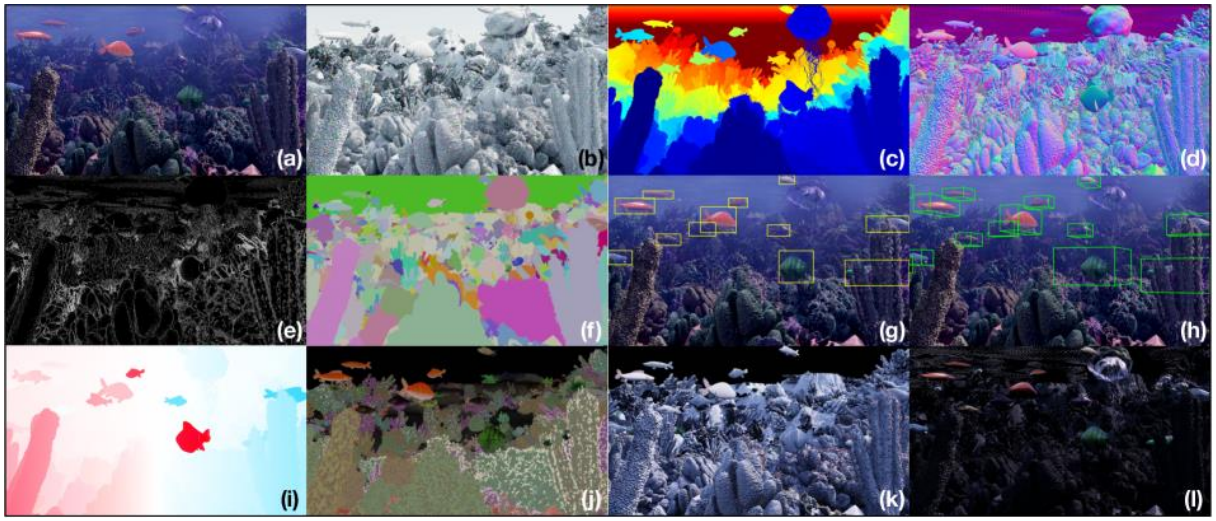


Figure 5: A set of images generated by Infinigen containing the main rendering image (a), and for it the high-res mesh (b), readily yields Depth (c), Surface Normals (d), Occlusion Boundaries (e), Instance Segmentation masks (f), 2D bounding boxes (g), 3Dbounding boxes (h). In addition, rendering metadata: Optical Flow (i), material parameters (j), Lighting Intensity (k) and Specular Reflection (l) are also included (Raistrick et al., 2023)

In recent years, the use of digital content creation (DCC) tools has gained increasing importance due to the ability to generate controlled labelled data. Digital Content Creation (DCC) generally refers to the creation of digital assets such as 2D/3D graphics, animation, video, sound, etc., and DCC tools are software programmes that help users to produce digital assets efficiently. Currently popular DCC tools are 3ds Max, Blender, Maya or After Effects. In a study of domain randomised 3D rendering to generate composite data, Blender was used to perform 3D rendering to produce high quality composite images (Tang & Jia, 2023). Compared to traditional DCC tools for creating and rendering high-quality static scenes, game engines are better at interactive real-time rendering. Game engines such as Unity and Unreal Engine are often used to create synthetic environments due to their real-time rendering capabilities and integrated physics engines that can simulate real physical interactions (Pollok et al., 2019). In this study of the use of synthetic data to improve the performance of target detection algorithms, the authors obtained diverse training data by varying the texture of the target, the camera height and angle, and the illumination

of the background in the Unreal engine (Damian et al., 2023). The Unity Perception package is an open-source toolset for generating synthetic datasets for CV tasks based on the Unity engine implementation. It extends Unity's editing components and functionality that allows Unity to directly generate annotated datasets (Borkman et al., 2021).

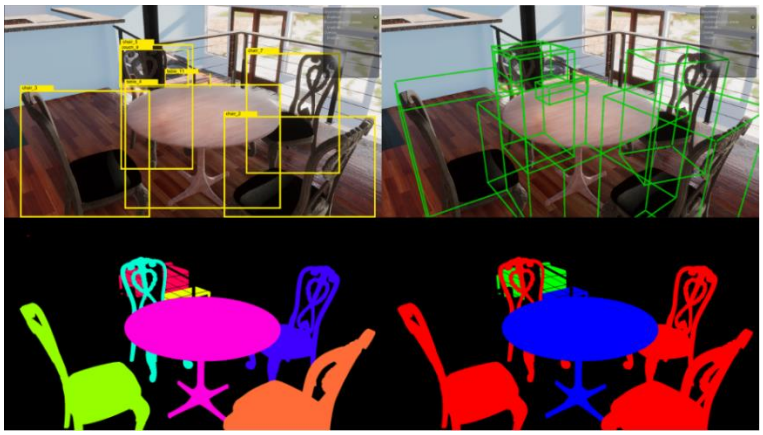


Figure 6: A set of images generated by Unity Perception, containing 2D Bounding Boxes (top left), 3D Bounding Boxes (top right), Instance Segmentation (bottom left), Semantic Segmentation (bottom right) (Borkman et al., 2021)

Simulators (e.g. Airsim or CARLA) are applications designed to create realistic environments for specific purposes, such as simulating vehicle movement in autonomous driving and drone flight. This is usually done in combination with a game engine, which provides customised parameters that allow the simulator to simulate the physical properties of a virtual space more closely to the real thing by means of a physics engine integrated with the game engine. Simulators can also be used to produce synthetic data, and this CARLA-based study has generated an instance-based and accurately labelled pedestrian dataset through an inverse projection pipeline (Lyssenko et al., n.d.).

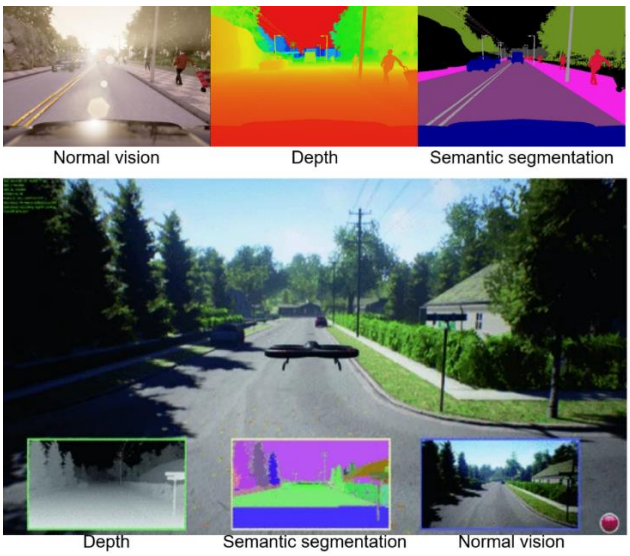


Figure 7: A set of images generated by Carla (top) (Dosovitskiy et al., 2017) and Airsim(bottom) (Shah et al., 2017)

Generative Adversarial Networks (GANs), on the other hand, are a newer approach to generating synthetic data. GANs consist of two neural networks, a generator that generates data samples that are close to the real one and a discriminator that determines whether the data is real or generated by the generator (Figueira & Vaz, 2022). After continuous adversarial training GANs can generate sufficiently realistic data that can enhance the diversity of the current dataset. Currently in the field of medical imaging, the datasets generated by GANs have been shown to improve the overfitting phenomenon caused by the lack of data (Eilertsen et al., 2021).

2.2.2. Scene Reconstruction

2.2.2.1. Traditional Methods

A 3D scene reconstructed from a 2D image is an important way to obtain synthetic data. Traditional 3D reconstruction methods are mature and widely used, and the key lies in acquiring depth information (Dawn & Biswas, 2019). They can be classified into active and passive methods (Aharchi & Ait Kbir, 2020). Active methods use a light source or active sensors to obtain depth information during the reconstruction process, such as structured light technology, which obtains depth information by projecting a light pattern onto an object to measure its surface geometry (Geng, 2011). TOF (Time of Flight) laser methods, which are commonly used in LIDAR systems, compute the depth information as a measurement of the time it takes for a laser pulse to reflect off the object and return to the sensor (Chua et al., 2016). In contrast, passive methods do not rely on active illumination and depth information is obtained through image features. For example, multi-view stereo vision (MVS) (Q. Zhu et al., 2021) techniques can calculate depth by analysing the geometric relationships between multiple camera views of the same object.(Furukawa & Ponce, 2010). In addition, another widely used method is Structure from Motion (SfM), which usually does not need to obtain the camera parameters in advance, and this method can simultaneously obtain the camera parameters and scene information data by analysing the feature points in the image sequence, which makes the SfM method less costly and has decimetre-level accuracy (Westoby et al., 2012). Thus, the SfM algorithm is widely used in UAV image processing and large-scale scene reconstruction. A study conducted in France used the SfM method to reconstruct the historic old town of the city of Bordeaux, with an error of less than 5 cm on the case of the Porte de Bourgogne reconstruction(Pepe et al., 2022).

2.2.2.2. Deep Learning Methods

With the development of deep learning, research and exploration of 3D reconstruction using deep learning methods has become increasingly active, although traditional reconstruction methods from 2D images to 3D are still dominant. Current popular approaches to 3D scene reconstruction using deep learning mainly include Convolutional Neural Networks (CNNs), Generative Adversarial Networks (GANs), and 3D reconstruction using implicit representations.

Convolutional Neural Networks (CNN) have been widely used in computer vision. In image processing, CNN has a significant advantage that it can directly use the image as input, obtain the features in the image to complete the matching in order to deduce the depth information and complete the

reconstruction of the 3D scene (H. Luo et al., 2024). Deep learning methods are not in conflict with traditional 3D reconstruction methods, on the contrary, the mutual participation of these two methods can improve the accuracy and efficiency of reconstruction from 2D images to 3D. For example, in this study of 3D scene reconstruction using the MSV method, CNNs are used to extract features from images at multiple scales and infer a depth map, which improves the efficiency of scene reconstruction and performs well in situations such as occlusion of targets and drastic changes in camera viewpoints (Abdullah, 2024). Another related study demonstrates the additional advantage of CNNs to generate scenes with semantic segmentation. By using incomplete 3D reconstructions and their corresponding labelled 2D RGB-D images for training, the model proposed in this study can generate high-quality reconstructed scenes with 3D semantic segmentation directly from 2D images without 3D annotations over the long training period (J. Huang et al., 2023).

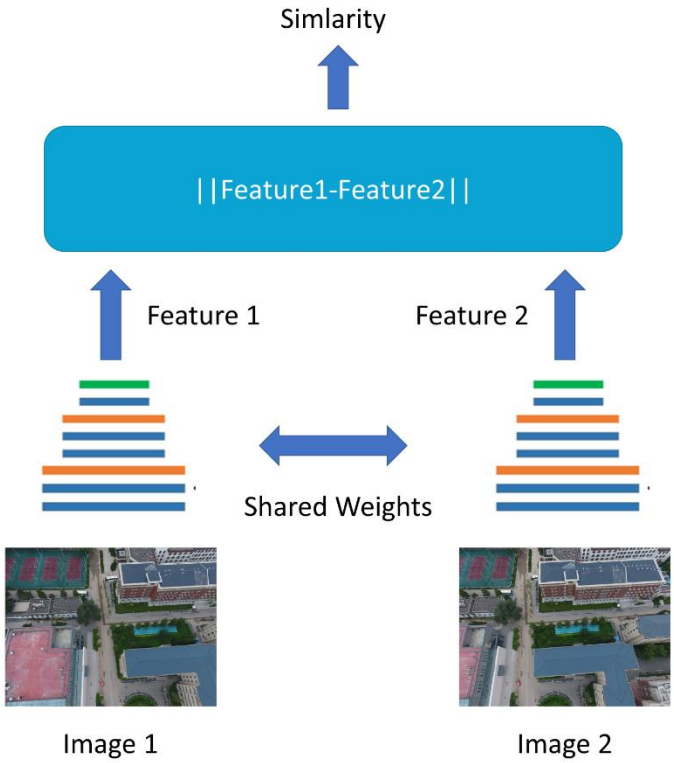


Figure 8: Architecture of Siamese networks (Khamis et al., 2018), where two CNNs with shared weight are used to extract features from stereo images(H. Luo et al., 2024)

Generative Adversarial Network (GAN) can output relatively high-quality 3D reconstruction results with its adversarial training strategy, that is., the generator completes the reconstruction from a 2D image to a 3D scene while the discriminator is responsible for evaluating the realism of the generated results (Samavati & Soryani, 2023). A notable example is GAN2Shape, an unsupervised 3D reconstruction method. It uses an ellipsoid as the initial image and renders pseudo-samples with unrefined shapes under multiple viewing angles and illumination conditions via a differentiable renderer. Next, the GAN is used to invert the original image corresponding to each pseudo-sample and the viewpoints and illumination conditions in which it is placed to optimise the 3D shape. After performing these steps iteratively, the final

GAN2Shape can generate fine-grained 3D shapes from a single image in an unsupervised situation (Pan et al., 2020).

Figure 9: GAN2Shape generates 3D shape from a single image (Pan et al., 2020)

3D reconstruction using an implicit representation treats the representation of the 3D scene as a continuous function rather than discrete, which allows the method to fit the function through a neural network to infer the desired reconstructed 3D shape. Currently popular implicit representations of 3D reconstruction are Neural Radiation Field (NeRF) with 3D Gaussian splatting. NeRF treats the 3D scene as a continuous volumetric field and obtains a free view of the 3D reconstructed target by using a deep neural network to map the 3D spatial coordinates obtained from camera ray sampling in a 2D image with the camera's viewpoint to RGB and density values. A study. demonstrated that NeRF can generate high-precision 3D reconstruction from coefficients of a 2D image scenes (Mildenhall et al., 2020). Different from NeRF, 3D Gaussian Splatting represents the whole scene by a series of consecutive ellipsoids with Gaussian distributions, each of which contains parameters such as position, colour and transparency, and these ellipsoids are rasterised to complete the rendering and to obtain a new view to complete the 3D reconstruction (Kerbl et al., 2023). In addition, 3D Gaussian Splatting can also be applied with outdoor large-scale scene reconstruction, for instance., GauU-Scene, a large-scale scene reconstruction benchmark of more than 1.5 km 2 done on the publicly available dataset U-Scene using 3D Gaussian Splatting, which confirms the effectiveness of the method for large-scale scenes (Xiong et al., 2024).

There are several well-established software packages that support the reconstruction of 3D scenes from 2D images, among which the popular ones using traditional methods are RealityCapture, AliceVision (Samavati & Soryani, 2023), 3DF Zephyr; and Lumi AI (NeRF), which uses deep learning methods.

2.2.3. Training Deep Learning Models with Synthetic Data

2.2.3.1. Related cases

The use of synthetic data for training deep learning models is starting to become commonplace, and the performance of synthetic datasets has been partially validated. For example, UnrealGT (Pollok et al., 2019), a research paper on generating ground truth datasets using the Unreal Engine, presents a framework for generating synthetic test data using the Unreal Engine. With synthetic data generation, large amounts of images and metadata can be extracted directly from a virtual scene, which can then be customised to meet the specific needs of an algorithm or use case. The paper evaluates their framework by generating synthetic test data and using that data to train and evaluate CNN as well as V-SLAM algorithms for target detection. The evaluation shows that the synthetic data they generate can be used as a substitute for real data, and it can be used in the fields of UAVs and autonomous vehicles. In addition, Sven Burdorf

et al. (Burdorf et al., 2022) also investigated the differences between deep learning models trained with synthetic data and those trained with real data. The study proposed a power-law-based evaluation method to quantify the actual amount of data that can be saved by using synthetic data from a mixed dataset. The results show there is no significant difference in performance between pre-training with fine-tuning and a training strategy that uses a composite dataset with a proportion of real data between 5% and 20%. Furthermore, this study shows that adding synthetic data can help improve object detection performance even with a small percentage of real data.

2.2.3.2. Existing Synthetic UAV Datasets

A substantial number of synthetic datasets have already been created and put into use. For example, "Mid-Air" (Fonder & Van Droogenbroeck, 2019) is a dataset that simulates UAV flight postures using Airsim and renders images through the Unreal Engine, and covers different seasons, weather and lighting conditions. Sim2Air (Barisic et al., 2022) is a publicly available synthetic dataset that can be used for UAV target detection, which uses Blender for model creation and rendering, with randomly assigned textures for reality-based shapes. In addition, another study (Xing & Tzes, 2023) proposed a diffusion model-based approach using text and masks as inputs to generate synthetic images via a diffusion model and specifically created a usable synthetic dataset.

Figure 10: The MidAir dataset contains data showing different climate settings. The top row shows the four simulated weather conditions. The bottom row illustrates the seasonal conditions (Fonder & Van Droogenbroeck, 2019)

3. METHODOLOGY

3.1. Dataset Description

Three datasets will be used in this study, VisDrone-Dataset, External dataset and Synthetic dataset, where Synthetic dataset will be collected in this study.

3.1.1. VisDrone-Dataset

VisDrone2019 dataset contains sub-databases for five different tasks: object detection in images; object detection in video; single-target tracking; and multi-target tracking with crowd counting. The entire database contains 288 video clips, 261,908 frames and 10,209 still images, as well as accurately annotated ground truth data. This research will mainly use the first sub dataset in order to train a deep learning model for object detection.

Figure 11: Example of the VisDrone dataset

3.1.2. External Dataset

External dataset is a collection of videos and images of the study area Road Boerderijweg with vehicle speed ground truths and bounding boxes; it contains 18 videos with ground truth vehicle speeds and 260 original images with vehicle bounding boxes. This dataset will be used primarily to test and examine the model performance.

Figure 12: Example of the External dataset

3.1.3. Synthetic Dataset

The synthetic data will be collected in the synthetic environment reconstructed in this study and is expected to contain approximately 10,000 images and their corresponding label files. This dataset will be primarily used to train and fine-tune the target detection capabilities of the deep learning model.

3.1.4. Study Area

Road Boerderijweg is the object to be reconstructed in this study. It is approximately 440 metres long and is located on the campus of the University of Twente, with a motorway in the centre and cycle lanes on both sides. The legal maximum travelling speed on this road is 50km/h. The synthetic dataset will be acquired from the scene reconstruction of this study area.

Figure 13: Study Area

3.2. Research Methods

This section highlights and discusses in five subsections the main steps of this study, as well as the methodological routes and software used for each step. The first subsection discusses the reconstruction of the synthetic environment; the second subsection discusses the acquisition of synthetic data; the third section discusses the training and fine-tuning of the deep learning model using real and synthetic data; the fourth section discusses the estimation of vehicle speed and compares the performance of deep learning models trained using different data.

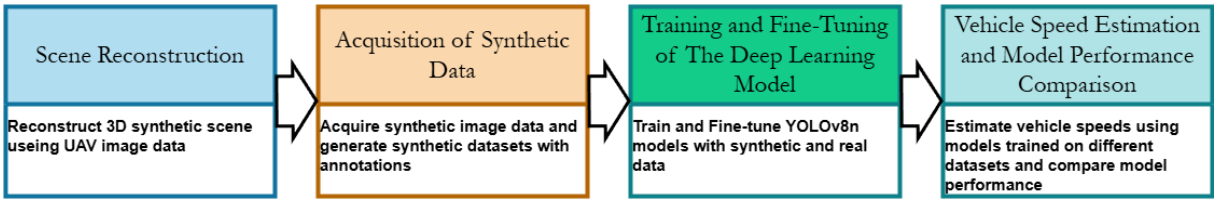


Figure 14: Overall Workflow

3.2.1. Scene reconstruction

Scene reconstruction is the first step in creating a synthetic environment for training deep learning models. The process involves acquiring aerial imagery, generating a 3D mesh and refining the mesh to address quality issues, as well as setting realistic lighting conditions and integrating dynamic actors (vehicles and vegetation) to further enhance the scene.

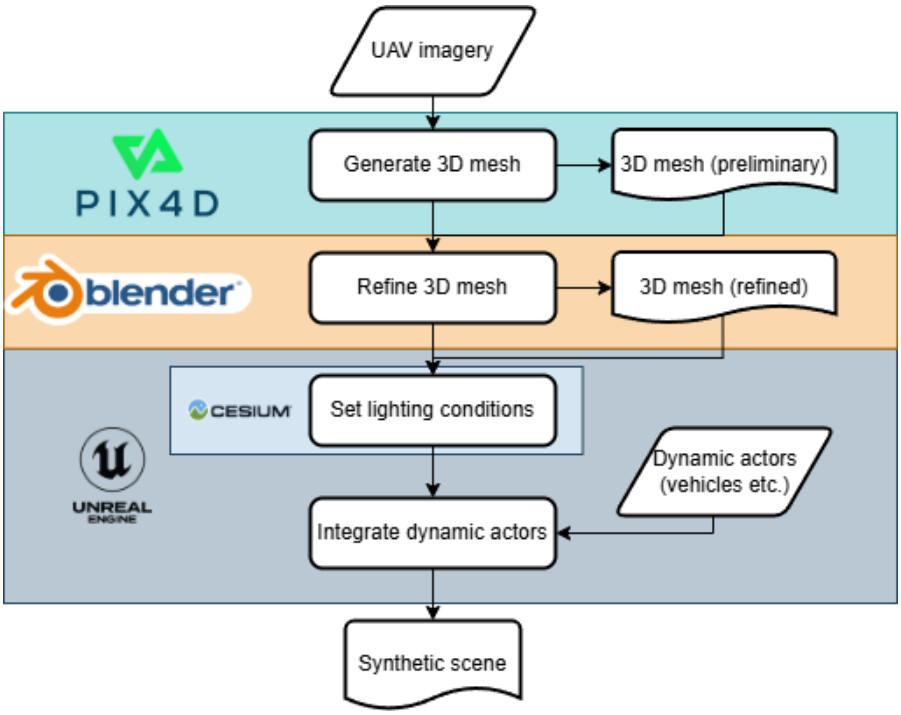


Figure 15: Scene Reconstruction Workflow

Images were captured using a DJI Phantom 4 Pro V2.0, with its high-resolution on-board camera that suitable for generating 3D meshes and texture. Since the study area can be considered as a rectangle,

double grid flight pattern was used, consisting of two vertically orthogonal flight paths within the planning area (study area) at an altitude of 50 metres, with 80% overlap between the images, to ensure full coverage of the study area (Torres-Sánchez et al., 2018).

PIX4D is a commercial photogrammetric software for reconstructing scenes from acquired images. Pix4D uses the structure from motion (SFM) to feature-matched camera positions and orientations based on overlapping images and then uses multi-view stereo (MVS) to generate a dense point cloud before generating a 3D mesh (Koutalakis et al., 2024). This SFM-based approach utilises multi-view geometry to accurately reconstruct the 3D structure of the scene (Firdaus & Rau, 2017). Its output (3D mesh with textures) provides the necessary content for the creation of the synthetic environment.

The study area, characterized by the presence of vegetation, buildings, asphalt pavement, and dynamic objects (e.g., vehicles, pedestrians), introducing occlusions, low-texture surfaces, and inconsistencies in the point cloud due to moving objects which in turn causes inconsistencies in the point cloud. These factors resulted in defects in the initial 3D mesh, such as holes, spikes, and overly smoothed regions (Haala et al., 2015). To address these issues, meshes and associated textures were imported into Blender, an open-source 3D modelling software with advanced editing capabilities. Using Blender's mesh editing tools, manual adjustments were made to fill in holes, remove spikes and correct areas of excessive smoothing to obtain a mesh that more accurately represents the study area.

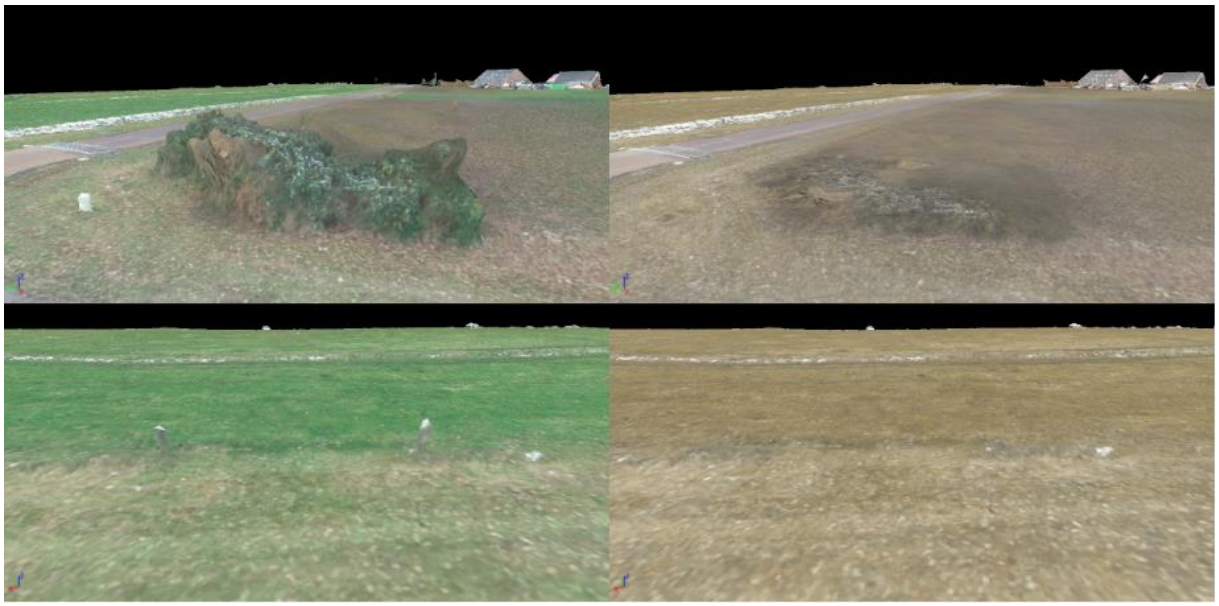


Figure 16: Example of manually adjusting a 3D mesh. On the left is the preliminary 3D mesh, on the right is the refined

The refined mesh was imported into Unreal Engine 5 (UE5) to complete the synthetic scene. Unreal 5 is an open-source game engine with a powerful rendering and physics engine that can be used to simulate drones and ground transportation systems (Shah et al., 2017). Unreal 5 is also geo-aligned, which means that it is possible to get real-world geographic coordinates in a composite environment. By adding the CesiumGeoreference actor to the scene, the CesiumGeoreference actor maps global geographic

coordinates based on WGS84 into Unreal Engine's local coordinate system. In this case, other assets in the scene will get a geographic coordinate that corresponds to the real world. In this project, Cesium for Unreal was used to provide scene lighting. This plugin for the Unreal 5 engine has a sophisticated location-based sunlight simulation, ideal for providing near-realistic lighting conditions for synthetic outdoor environments. Use the CesiumSunSky function under the Cesium plugin to set the lighting to simulate 9 April 2024 at 1500 hours (DST) and adjust the shadows to reflect real-world conditions. Vegetation and vehicles were added to the scene using UE5's asset library, adding a total of 31 vehicles. The vehicles were animated using the blueprint system so that they could follow preset routes on the roads in the scene to simulate a realistic scene suitable for acquiring deep learning training data.

3.2.2. Acquisition of Synthetic Data

Obtaining a synthetic dataset is a critical step in accomplishing deep learning model training. This mainly includes acquiring the image data from the built synthetic environment and completing the bounding box annotation of the image data.

3.2.2.1. Image Data Capture

In order to simulate the images obtained from the flight of an unmanned aerial vehicle (UAV), the camera character is integrated into the compositing environment developed in Unreal Engine 5 (UE5). The camera is mounted on a slide component and is set to traverse a predefined trajectory that simulates the UAV's flight path. the Movie Render Queue (MRQ) plug-in in UE5 is used to record video from the camera's point of view. the MRQ is configured to render two simultaneous pipelines: the main RGB pipeline, which captures the scene inside the camera's view cone, and a custom rendering pipeline, which generates a mask specific to the vehicle by isolating it from the background. vehicle-specific masks by isolating the vehicle from the background. This two-channel setup ensures pixel-level correspondence between RGB frames and masked frames.

A custom stencil layer was activated in the UE5 project to enable vehicle masking. Each vehicle role is assigned a unique stencil value (1 to 40) that is rendered as a different colour in the custom channel, excluding background elements. In order to support subsequent bounding box extraction, vehicle masks require highly distinguishable colours. Assuming a scene with up to 40 vehicles, 40 unique colours need to be defined in the HSV colour space with constraints on hue (H: 0-255), saturation (S: 0.75-1.0) and value (V: 0.75-1.0). The greedy algorithm maximises the Euclidean distance between these colour points, thus ensuring separation and minimising overlap (Glasbey et al., 2007). The generated HSV values are mapped to stencil values and rendered as RGB colours in a custom channel. MRQ then renders the RGB and mask pipelines using a synchronised sequence of frames, ensuring a one-to-one correspondence between the frames of the two pipelines.

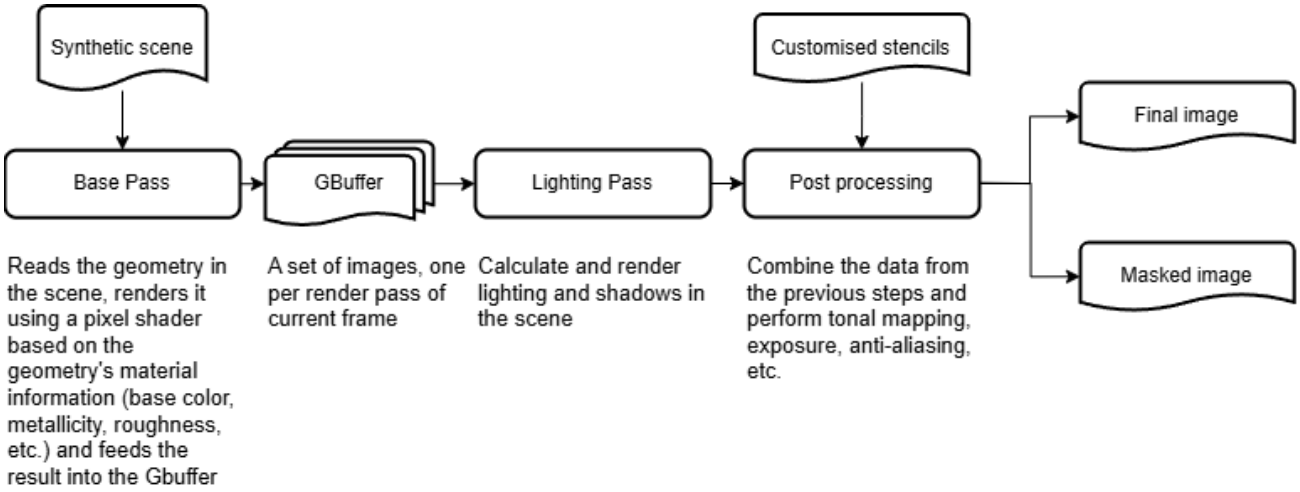


Figure 17: The generally Rendering Pipeline used in this study

3.2.2.2. Bounding Box Generation

The bounding box annotations were generated using Python and the OpenCV library, which provides powerful contour detection and bounding box functions that are well suited for high contrast masked images. However, since UE5 is unable to fully separate the main render pipeline from the custom render pipeline, this has resulted in post-processing effects of the main pipeline being introduced into the custom pipeline, which has resulted in noise in the colours of the vehicle masks in the masked images and the number of vehicles (and the resulting unique colours) differing from frame to frame. These issues need to be resolved by quantifying the colours in the masked images.

The colour quantification of the masked images is implemented by means of the k-means clustering algorithm (Thompson et al., 2020). First, the number of unique vehicle colours is determined by converting each masked image (UE5 outputs images in RGB colour space) to HSV colour space, where the H-channel mainly affects colour differentiation. The histogram of H-values is calculated, and the number of different peaks plus 1 peak for the background estimates the total number of unique colours. This value served as the k parameter for the k-means clustering algorithm, which is able to accommodate different numbers of vehicles in different frames. K-means quantifies the image colours, reclassifying similar colours in the masked image into a uniform colour to match the number of vehicles, ensuring that each vehicle is represented by a single colour. Separate masks are created for each colour (each vehicle) identified by k-means in addition to the background, and OpenCV's morphological operations are used to filter out noise and connect nearby blocks of colour representing the same vehicle.

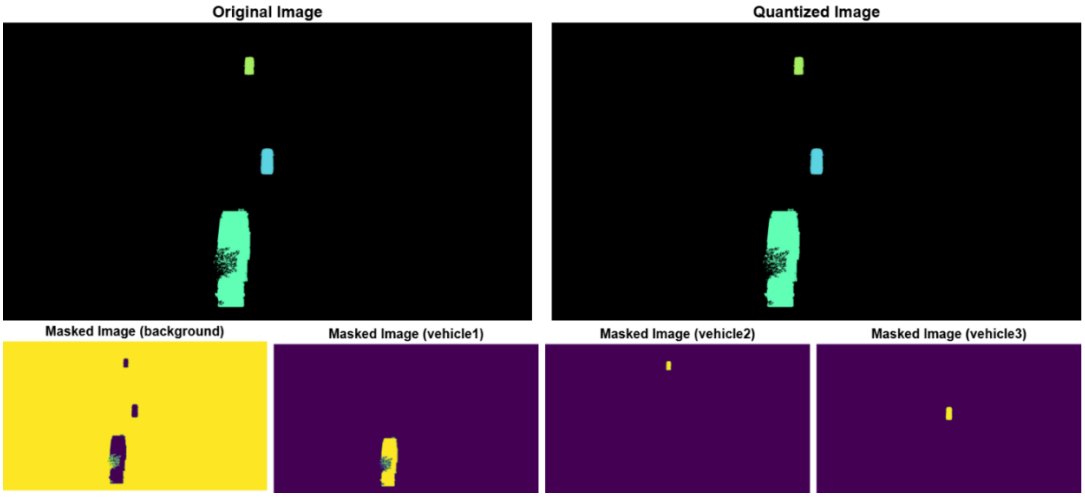


Figure 18: Example of color quantization

The contours are detected in a separate mask for each vehicle using OpenCV's findContours function, which saves information about the contours' position in the image in YOLO format and ensures that they are paired with their respective synthetic RGB frames.

3.2.3. Training and Fine-Tuning of The Deep Learning Model

In this study, YOLOv8n (You Only Look Once, version 8 nano), the smallest and fastest version of the YOLOv8 family, was selected for its efficiency on devices with limited computational resources, such as UAVs (Terven & Cordova-Esparza, 2023). Two datasets were used: the VisDrone dataset, an open-source collection of UAV-captured images, and a custom synthetic dataset, as detailed in Section 4.2. For each dataset, the YOLOv8n model was trained from scratch. Training was performed on a laptop with an NVIDIA RTX 3060 graphics card, using a batch size of 16, a learning rate of 0.01, and running 100 epochs per dataset.

Subsequently, fine-tuning was performed by training each model on an alternate dataset: models initially trained on VisDrone were fine-tuned using the synthetic dataset and vice versa. The training parameters were kept constant. All four generated models, two from initial training (VisDrone training and synthetic training) and two from fine-tuning (VisDrone to synthetic and synthetic to VisDrone) were saved for the following performance evaluation.

3.2.4. Vehicle Speed Estimation and Model Performance Comparison

This section describes the methodology for estimating vehicle speeds using a pipeline that integrates vehicle detection, tracking and speed calculation. Four YOLOv8n models, trained and fine-tuned as described in Section 3.2.2, are used with the ByteTrack algorithm to process the test video, which enables performance comparisons to be made with known vehicle speeds.

The vehicle speed estimation pipeline consists of three interrelated components: vehicle detection, tracking and speed calculation. Vehicle detection is performed using four YOLOv8n models, each trained

and fine-tuned on a combination of the VisDrone and Synthetic datasets, as described in Section 3.2.3. Tracking is implemented using ByteTrack, a multi-target tracking algorithm that employs Kalman filtering to predict target trajectories and uses the Hungarian algorithm to correlate detections with trajectories (Y. Zhang et al., 2022). Unlike conventional trackers that prioritise high-confidence detection boxes, ByteTrack associates all detection boxes, enhancing the robustness of low-score detection. Speed estimates are derived from positional changes between frames.

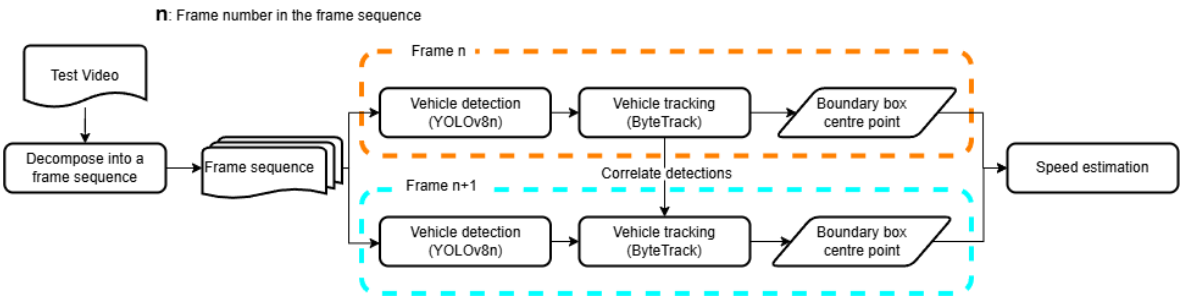


Figure 19: Vehicle Speed Estimation workflow

The test video, taken from an external database, depicts a vehicle travelling from east to west at a constant speed of 15km/h, captured by a UAV flying in the same direction at a fixed altitude of 50 metres above the vehicle and recorded at 30 frames per second (fps). The trained and fine-tuned YOLOv8n model processes each frame individually to detect the vehicle, and the resulting bounding box is fed into ByteTrack for tracking throughout the sequence.

Vehicle speed estimation is achieved by analysing the positional displacement between successive frames. For each detected vehicle, the geometric centre of its bounding box is represented as the vehicle's position in the current frame, mathematically represented as:

Equation 1

$$Current_{centre} = \left(\frac{x_0 + x_1}{2}, \frac{y_0 + y_1}{2}\right)$$

where (x_0, y_0) represents the top-left corner and (x_1, y_1) represents the bottom-right corner of the bounding box. The pixel displacement is then determined by calculating the Euclidean distance between the centres of the bounding boxes (P_d) in consecutive frames:

Equation 2

$$P_d = \sqrt{(x_c - x_p)^2 + (y_c - y_p)^2}$$

Where (x_c, y_c), (x_p, y_p) denote the centre of detection bounding box of the current and previous frame, respectively. It should be noted that YOLOv8n uses an image format in which the pixel coordinate origin of the image is located in the upper left corner of the image, which also means that when the centre of the detection bounding box is shifted horizontally to the left relative to the previous frame, the value of P_d is negative. Given the fixed altitude and orthogonal view of the UAV, the Ground Sampling Distance (GSD) is derived from the camera's sensor size, focal length, and flight altitude and is determined to be 0.016519 metres per pixel. The actual displacement between frames is calculated by multiplying the pixel distance by the GSD, and the vehicle speed (V) in metres per second is calculated as:

Equation 3

$$V = U - \left(\frac{P_d \times GSD}{f}\right) \times F$$

where f is the frame interval, which is the frame difference between the two detected frames; GSD is the ground sampling distance; F is the video frame rate; and U is the UAV speed.

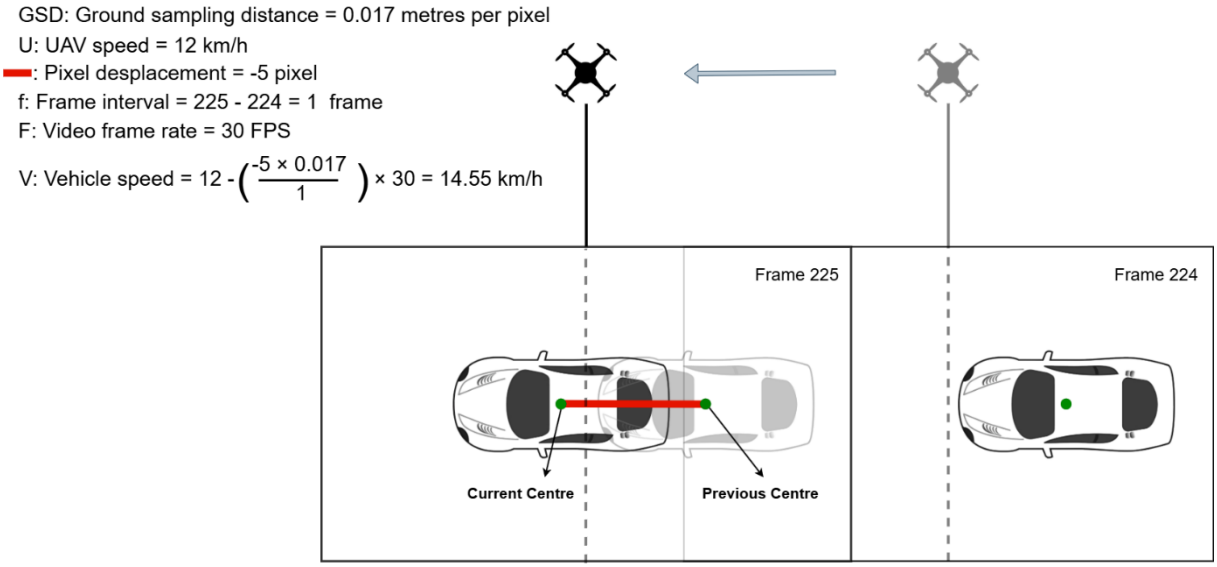


Figure 20: Schematic diagram of how to calculate the speed of a vehicle, the values of the parameters are fictitious.

Each of the four YOLOv8n models (VisDrone training, synthetic training, VisDrone to synthetic fine-tuning, and synthetic to VisDrone fine-tuning) was independently integrated into the pipeline. The estimated speeds of each model were compared to known vehicle speeds to evaluate their performance.

4. RESULT

This section presents the results of the study and is divided into four subsections corresponding to the methodological steps outlined in the Research Methodology section. These subsections detail the results of scene reconstruction, synthetic data acquisition, deep learning model training and fine-tuning, and vehicle speed estimation and model performance comparison.

4.1. Scene reconstruction

The result of the scene reconstruction is an Unreal 5 (UE5) project containing a mesh body of the study area reconstructed by Pix4D from aerial images captured by DJI Phantom 4 Pro V2.0 and refined by Blender, 31 animated vehicles and vegetation from the UE5 asset library. This project also integrates the cesium plugin to freely adjust solar lighting to reflect real world conditions. Figure 10 provides a rendering of the final synthesised scene, demonstrating the realistic integration of the dynamic Actor and lighting, suitable for generating synthesis training data.

Figure 21: Rendering of a composite scene in an Unreal Engine project

4.2. Acquisition of synthetic data

A synthetic dataset generated from the UE5 environment that captures changes in vehicle position, lighting conditions and camera angles. It contains 10,172 training sets, 1,396 validation sets and 2,759 test sets, totalling 14,327 images with a resolution of 1920x1080 pixels. All images are labelled with their corresponding yolo format labels, as well as RGB masks with drawn bounding boxes for rendering. Figure 11 shows a sample RGB image and a sample mask image demonstrating the clear separation of the vehicle from the background.

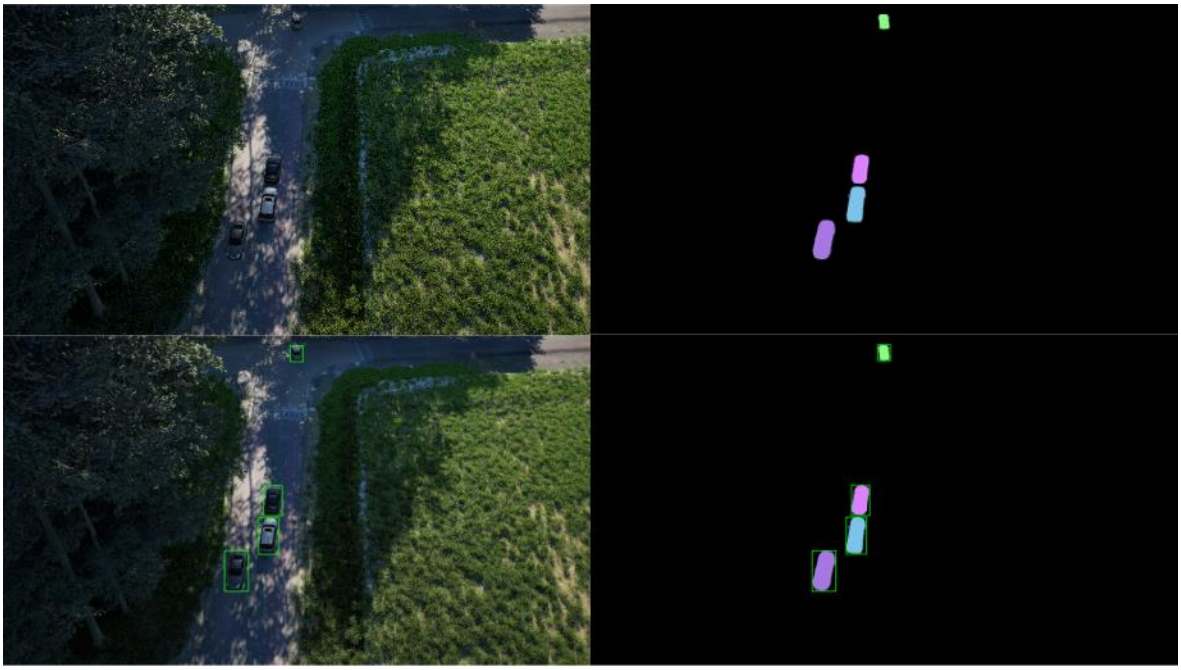


Figure 22: Final image (top left), masked image (top right), Final image with bounding box (bottom right), masked image with bounding box (bottom right)

4.3. Training and Fine-Tuning of The Deep Learning Model

The performance of the YOLOv8n model was evaluated with four sets of controlled experiments (train43-train46) with different training strategies. These models include Train43, which was trained from scratch using a custom synthetic dataset; Train44, which was trained from scratch using the VisDrone dataset containing real UAV images; Train45, which was pre-trained using the synthetic dataset and fine-tuned on the VisDrone dataset; and Train46, which was pre-trained using the VisDrone dataset and fine-tuned on the VisDrone dataset pre-training and fine-tuning on the synthetic dataset. All models are trained for 100 epochs on NVIDIA RTX 3060 GPUs with an image size of 640×640 pixels and a learning rate of 0.000298. Table 1 summarises the validation metrics, including the precision and recall for IoU thresholds of 0.5 (mAP50) and 0.5 to 0.95 (mAP50-95).

Table 1: Validation Metrics for YOLOv8n Model

Model	mAP50	mAP50-95	precision	recall	train/box_loss	train/cls_loss
train43	0.96	0.87	0.99	0.92	0.37	0.22
train44	0.77	0.51	0.86	0.67	1.1	0.58
train45	0.77	0.51	0.87	0.68	1.07	0.56
train46	0.96	0.87	0.99	0.92	0.35	0.21

Train43 has a mAP50 of 0.960 and a mAP50-95 of 0.872, with a precision and recall of 0.987 and 0.915, respectively, which is significantly higher than that of Train44 (mAP50 of 0.766 and mAP50-95 of 0.506, with a precision and recall of 0.861 and 0.670, respectively). The synthetic model (Train43) maintained high precision at all confidence thresholds, whereas the VisDrone-trained model (Train44) showed a larger

decrease in precision at higher confidence levels. This reflects the robust detection performance of the model on the synthetic dataset in the restricted region, while the model trained from scratch on the real dataset (Train44) exhibits relatively poor performance due to the complexity of the real-world data. Fine-tuning improves performance: Train45 is fine-tuned on VisDrone after synthetic pre-training, and mAP50 reaches 0.771 and mAP50-95 reaches 0.512, which is slightly higher than Train44, with precision and recall improving to 0.866 and 0.676. Train46, fine-tuned on the synthetic dataset after VisDrone pre-training, maintained an mAP50 of 0.960 and improved mAP50-95 to 0.874, with precision and recall of 0.990 and 0.916. The precision-confidence and mAP50 with mAP50-95 curves are displayed as visualised in Figure 12. These results suggest that pre-training with synthetic data followed by fine-tuning using real data improves the performance of real-world detection (Nowruzi et al., 2019). Correspondingly, fine-tuning the real data model using synthetic data maintains high accuracy in the synthetic environment.

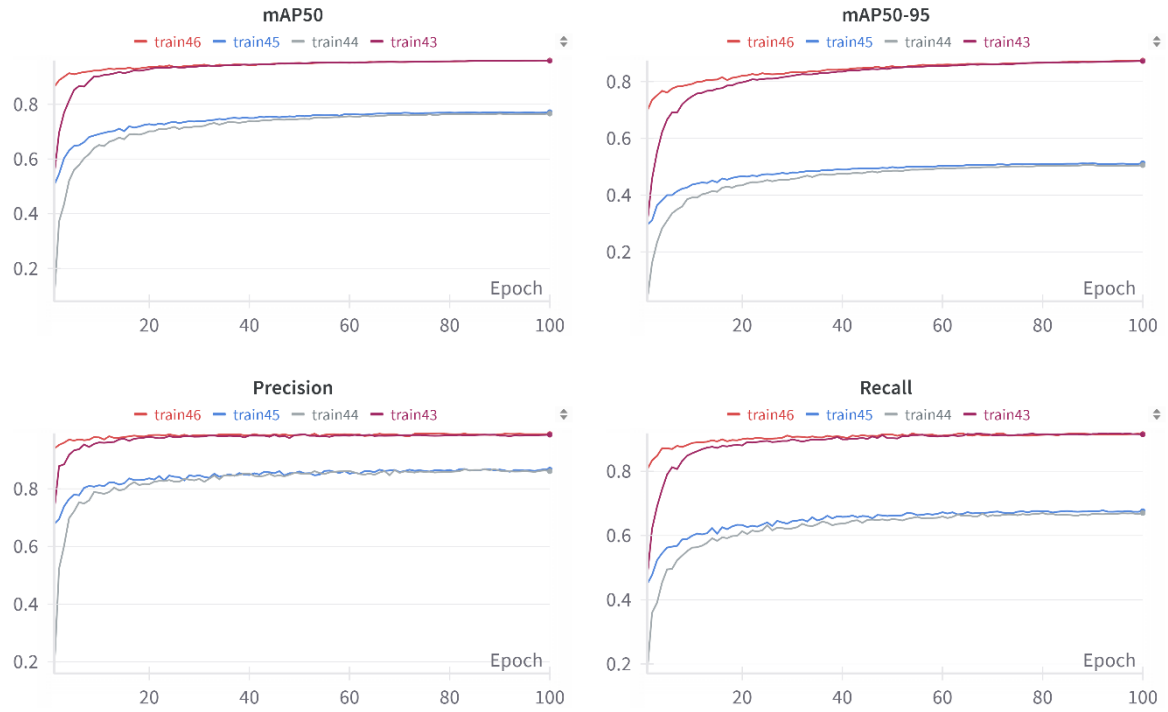


Figure 23: mAP50 with mAP50-95 curves and Precision-confidence curves

4.4. Vehicle Speed Estimation and Model Performance Comparison

The speed estimation performance of the four YOLOv8n models was evaluated on a test video in an external dataset depicting a vehicle that is expected to travel at approximately 15 km/h, captured by a UAV travelling in the same direction as the vehicle, at a fixed altitude of 50 metres and at a frame rate of 30 frames per second (fps).

Table 2 summarises the performance comparison results of the four models. As a special note, the truth (ground truth) speeds of all models are computed based on the same video frames as the speed estimation task to ensure data source consistency.

Table 2: Performance comparison results for the four models

Model	Training Scenario	MSE	RMSE	MAE	Average speed (Estimation)	Average speed (Ground Truth)	Estimated deviation rate
Train 43	Synthetic only	26.72	5.17	3.82	14.19	17.36	-18.27%
Train 44	VisDrone only	29.94	5.47	4.17	13.46	17.19	-21.68%
Train 45	Synthetic → VisDrone	27.60	5.25	3.99	13.68	17.18	-20.40%
Train 46	VisDrone → Synthetic	26.37	5.13	3.81	13.99	17.20	-18.67%

The results of the model performance comparisons showed that all models systematically underestimated the true speed, with a mean estimate ranging from 13.46 to 14.19 compared to the true speed of approximately 17.03 km/h. Among the four training scenarios, the model pre-trained with VisDrone and fine-tuned based on synthetic data (Train46) had the lowest Mean Absolute Error (MAE) of 3.809, closely followed by the model trained using only synthetic data (Train43) with a MAE of 3.8156. The model trained only based on VisDrone data (Train44) has the highest MAE of 4.1737, while the model pre-trained based on synthetic data and fine-tuned based on VisDrone data (Train45) has an intermediate MAE of 3.9868. The RMSE with estimated speeds reflects the same results. Train46 performs the best, having the lowest REMS value and the speed estimate closest to the ground truth, Train43 and Train45 follow in that order, while Train44 performs the worst, with an RMSE of 5.4719 (the highest) and the largest difference between the speed estimate and the ground truth value. The difference in speed estimation performance of the models generally matches the difference in target detection performance of the models in the previous section.

5. DISSCUSION

This chapter discusses the performance of the 4 models with the limitations of the vehicle speed estimation study, and potential improvements.

5.1. Limitation

This section discusses the limitations of the methodology and results of the speed estimation study and the possible causes of these problems.

5.1.1. Systematic errors in speed estimation results

In the speed estimation experiments, all the models tended to underestimate the average speed of the vehicles in the test videos, and it is possible that this systematic error is due to the occlusion of the targets in the test environment, which significantly affects the accuracy of the pixel displacement (P_d). In the speed estimation experiments, all the models tended to underestimate the average speed of the vehicles in the test videos, and it is possible that this systematic error is due to the occlusion of the targets in the test environment, which significantly affects the accuracy of the pixel displacement (P_d). The velocity formula： $V = U - \left(\frac{P_d \times GSD}{f}\right) \times F$ used in this study has a velocity （V） that is very susceptible to the P_d value, which is affected by the accuracy of the detection bounding box. In the test scenario, the vehicle is travelling from east to west and the UAV is flying parallel to the vehicle in the same direction. Ideally, the target vehicle is approximately positioned in the centre of the frame and produces a small horizontal displacement to reflect its motion relative to the UAV. This means that the P_d value is also theoretically very small most of the time. However, some of the roads in the test video were affected by vegetation occlusion, and the target detection frame could not accurately fit the target vehicle contour. The figure below shows the detection frame and the change of its centre point position due to the occlusion. As shown in the figure, the occlusion of vegetation leads to an unexpected vertical shift of the centre point of the target detection frame between frames.

Figure 24: Vegetation occlusion causes shift in the centre point of the target detection edit box

Since the application of this formula assumes that the trajectories of the target vehicle and the UAV are parallel, changes in the vertical displacement of the centroid are incorrectly assumed by this formula to be in the direction of the vehicle's forward motion, which is the change in the horizontal direction of the centroid, leading to an inflated P_d value and consequently a lower velocity estimate V. These underestimated velocities are ultimately reflected in the average (estimated) speed, resulting in a systematically low estimate of V.

5.1.2. Inaccurate Speed Estimation Results

The accuracy of the speed estimates was also lacking, with Mean Absolute Errors (MAE) ranging from 3.81 to 4.17 and Root Mean Square Errors (RMSE) ranging from 5.13 to 5.47. This could be caused by tracking errors. The velocity estimation pipeline used in this study relying on ByteTrack for target tracking, which uses Kalman filtering and the Hungarian algorithm to correlate detection frames across frames. However, in the presence of occlusions, small targets, or fast-moving vehicles, ByteTrack's performance may degrade, resulting in misassociations or lost trajectories (Y. Zhang et al., 2022). Such errors directly affect the calculation of the P_d value, leading to inaccurate speed estimation. Inaccurate velocity estimation caused by tracking problems is also a common problem in UAV-based target velocity estimation. In addition to this, the calibration of the Ground Sampling Distance (GSD) also introduces errors. The conversion of pixel displacement to actual distance depends on an accurate GSD, and since this experiment assumes that the height of the UAV is constant, a fixed GSD value (1.34146 cm/px) is used. However, any small change in the UAV altitude and potentially lens distortion introduces systematic errors and leads to larger errors in the derived metric of velocity estimation. Fixed GSD values cannot cope with this challenge, while using accurate GSD calibration can minimise the error that occurs when converting pixel displacements to actual distances, and thus ensure the accuracy of speed estimation (J. Li et al., 2019). Model architectural constraints may be another reason affecting the performance of speed estimation. YOLOv8n is a lightweight model in the YOLOv8 family optimised for efficiency, and although it may be suitable for edge devices with limited computing resources, its ability to accurately detect targets in complex environments is at a disadvantage compared to the other models in the family

(Sohan et al., 2024). For detection of moving vehicles in complex environments, especially in occluded environments, more robust architectures may be required.

5.1.3. Dataset Design and Its Influence on Model Performance

The design of the synthetic dataset may have introduced bias into the speed estimation results. Models trained or fine-tuned on synthetic data (e.g., Train43 and Train46) performed well in both speed estimation and detection metrics, possibly due to the automatically generated, highly consistent bounding box of the dataset. Since the speed estimation algorithm in this study is calculated based on the cross-frame displacement of the centroid of the target's detection box, and the consistency of the detection box is important for maintaining target identification during tracking, this directly affects the accuracy of the speed estimation (Wojke et al., 2018). On the contrary, real datasets like VisDrone may introduce bounding box errors in the training set due to manual labelling, which may explain why Train44 (VisDrone only) versus Train45 (synthetic dataset pre-training, VisDrone fine-tuning) did not take the advantage in terms of detection and speed estimation. In addition to this, although the synthetic dataset is not real, its scenarios are reality-based and highly similar to the test data, which carries a potential risk of overfitting.

5.2. Improvements

This section focuses on several possible improvements proposed to address the limitations mentioned in the previous section.

5.2.1. Reduce Systematic Errors

In the previous section it was considered that the main reason for the general underestimation of the speed of the target vehicle by the four models was the unexpected vertical displacement of the centre point of the target detection box due to occlusion. A potential solution is to use only the horizontal component of the pixel displacement. Since the UAV can be considered to be flying horizontally parallel to the target vehicle in the test conditions of this study, discarding the displacement on the vertical component of all frames avoids the effect of vertical offset due to occlusion, thus providing a more accurate speed estimate. Another possible approach is to compute the displacement vector of the target vehicle. By calculating the change in the position of the centre point of the target vehicle in successive frames, the vector of the current frame is obtained, which in turn can approximate the forward direction of the target vehicle. Accordingly, the displacement of the target vehicle relative to the UAV in the forward direction can be known, avoiding the risk of vertical drift of the centre point and obtaining more accurate data.

5.2.2. Improve Speed Estimation Accuracy

As mentioned in Section 5.1.2, the performance of ByteTrack may be degraded in the presence of occlusion, small targets, or fast-moving vehicles, and replacing ByteTrack with a tracker that performs

better in occlusion conditions, such as Deepsort, is an option. Compared with ByteTrack, DeepSort can better deal with occlusion environments, and it can handle the problem of target occlusion and tracking breaks by fusing the appearance features, re-identifying and recovering the target trajectory after the occlusion or the brief loss of the target, which improves the continuity and robustness of tracking (Wojke et al., 2018). In addition, projects combining DeepSort with YOLO models as well as are widely used (Bin Zuraimi & Kamaru Zaman, 2021). The use of dynamic, precisely calibrated GSDs instead of fixed GSDs can also improve the accuracy of vehicle speed estimation. Using a UAV-mounted sensor to obtain its real-time attitude and coordinates, and calculating the GSD value accordingly, can reduce the error introduced by changes in the UAV's altitude and attitude to a certain extent. Replacing the YOLOV8n with a more powerful model from the same family, for instance, YOLOv8m or YOLOv8l, could also potentially improve the accuracy of speed estimation by improving the accuracy of target detection.

5.2.3. Model Training and Dataset Design Improvements

For the VisDrone dataset, data enhancement and preprocessing can be performed on the dataset during training, such as random cropping, expanding, horizontal flipping, random scaling, etc. as well as colour dithering (brightness, contrast, saturation adjustment). Such processing can enhance the recognition ability of the single-stage detector (YOLOV8n) for complex backgrounds and small targets (Z. Zhang et al., 2019). For synthetic datasets, to avoid the potential risk of overfitting, data generated in other synthetic scenes can be introduced to increase the diversity of the training data.

6. CONCLUSION AND RECOMMENDATION

6.1. Conclusion

This study develops a practical pipeline for semi-automatically reconstructing realistic outdoor scenes and generating synthetic images with annotations. Image data were acquired using a UAV, and a highly realistic scene was created by using a combination of well-established commercial and open-source software: Pix4D, for the preliminary generation of high-resolution 3D mesh; Blender, for the manual refinement of the mesh to correct for imperfections (holes, spikes); and Unreal Engine 5, for the combination of 3D mesh and dynamic actors, as well as the application of geo-referenced lighting conditions to create a near-realistic environment. In this synthetic scene, vehicles are placed along the road and animated to simulate real traffic situations. The image data was captured by a virtual camera simulating a UAV flight along a predetermined flight path. A two-channel rendering process was used: an RGB channel for the scene and a mask channel to set a unique colour for each vehicle. This setup ensures pixel-level correspondence between the RGB frames and the vehicle masks. A contour detection procedure using OpenCV is then performed to automatically extract bounding box annotations from the mask channels. In summary, a suitable scene reconstruction method was found and based on this a synthetic dataset (over 10,000 images) was generated which contains accurate, pre-annotated bounding boxes for each vehicle.

The results show that synthetic data can significantly improve UAV-based vehicle speed estimation models. We trained four YOLOv8n detection models under different scenarios: (1) using only synthetic data; (2) using only VisDrone (real) data; (3) synthetic pre-training followed by fine-tuning using VisDrone; and (4) VisDrone pre-training followed by synthetic fine-tuning. The model trained on synthetic data (Train43) achieves very high detection accuracy (mAP50 $\approx$ 0.96), with precision and recall of about 0.99/0.92, which is much higher than that of the model based on real data (Train44, mAP50 $\approx$ 0.77). Notably, the same level of detection performance was achieved by a model pre-trained on real images after fine-tuning the synthetic data (Train46, mAP50 $\approx$ 0.96). The results were similar when it came to vehicle speed estimation on real UAV test videos: the Train46 model (VisDrone$\rightarrow$synthesis) performed the best, achieving the lowest mean absolute error (MAE $\approx$ 3.81m/s), closely followed by the synthetic-only model (MAE $\approx$ 3.82m/s). In comparison, the VisDrone-only model had a MAE $\approx$ 4.17 m/s. Thus, fine-tuning using synthetic data reduces the velocity estimation error by approximately 10%. These findings confirm that training and fine-tuning with synthetic datasets can well improve the performance of deep learning models for vehicle speed estimation.

Overall, both research objectives established were achieved. A semi-automated workflow was developed to reconstruct a realistic outdoor scene and generate a synthetic dataset with accurate labelling. This dataset improves the performance of the deep learning model; training and fine-tuning using the synthetic dataset has higher detection accuracy and more accurate speed estimation. Although the synthetic environment is a highly simplified representation of the real scenario, the synthetic scene is still highly matched to the test scenario, and thus there is a potential risk of overfitting. Nonetheless, the current

findings still indicate that synthetic data can effectively enhance the performance of vehicle speed estimation algorithms.

6.2. Research Questions & Answers

1.1 What is the suitable method for the semi-automatic reconstruction of the scenarios in this study?
In this study, the synthetic scene reconstruction of the study area was reached by combining the use of UAV imagery data with well-established commercial and open-source software. This semi-automated approach (photogrammetry - manual editing - game engine simulation) proved to be effective: it allowed a fast and cost-effective reconstruction, keeping the geometry of the real scene while allowing fine-adjustments and dynamic elements. The final synthetic environment reproduces the real road and supports data generation. In summary, the appropriate approach is to generate a 3D mesh using Pix4D, then perform manual mesh correction in Blender and simulate it in UE5.

1.2 How to acquire data with pre- annotated in the reconstructed scene?
Acquire synthetic images by applying a virtual camera in UE5 that moves on a preset track and captures synchronised RGB and split-mask images using the Movie Render Queue. This two-channel rendering ensures that the RGB image and the corresponding colour mask image are captured in parallel for each frame. To generate the annotations, the mask images were post-processed using Python: K-mean clustering was applied to merge the vehicle colours, and OpenCV outline detection was used to extract a tight bounding box around each vehicle. This approach ensures that each composite frame is paired with an accurate object label. Through this process, this study generated over 10,000 images with bounding box labels in the reconstructed scene.

2.1 Can deep learning models trained and fine-tuned with synthetic datasets be generalised to real-world scenarios in the vehicle speed estimation task?
This was evaluated and tested by training the YOLOv8n detector under four scenarios (synthetic only, real only (VisDrone), synthetic → real fine-tuning, real → synthetic fine-tuning) and then going through a consistent speed estimation process on real UAV speed videos. The results show that the model using synthetic generalises well to real test environments, and that the model pre-trained with synthetic data is more accurate than the model trained with only realistic data.

2.2 To what extent will vehicle speed estimation performance be improved after fine-tuning using synthetic data?
The performance of the speed estimation was improved by fine-tuning using synthetic data. The mean absolute error was reduced from 4.17 km/h to 3.81 km/h, a reduction of 0.36 km/h (about 8.7%) in absolute terms. The relative error was also reduced from -21.73% to -18.70%, a reduction of 3.03%. These improvements prove that the synthetic training data enhances the estimation accuracy of the modes. Overall, the performance of vehicle speed estimation was improved by about 8%.

6.3. Recommendations for future work

Based on the results of the current study, there are the following recommendations to address the limitations of the study and extend its applicability. A much-needed improvement is to reduce the occlusion-induced errors in speed estimation by adjusting the calculation of pixel displacements. For example, calculating a displacement vector centred on the vehicle, or using only the horizontal component of pixel displacement to avoid vertical bias caused by foliage occlusion. It is also recommended to upgrade the tracking component: replacing ByteTrack with a more powerful tracker such as DeepSort (which incorporates appearance features and re-identification) can reduce identity switching due to tracking loss in case of occlusion. Additionally, implementing dynamic GSD calibration using real-time UAV attitude data to adjust the pixel-to-distance conversion can also improve the accuracy of velocity estimation, especially in the case of changes in UAV altitude or attitude. In terms of synthetic data, in order to prevent overfitting, future work should focus on extending the scene diversity by introducing more variations in lighting, weather, texture and traffic situations, which will help the model to generalise to different environments. Finally, experimenting with larger detection models (YOLOv8m or YOLOv8l) to further improve detection accuracy and thus speed estimation accuracy. By exploring these directions, subsequent research could enhance the quality of synthetic datasets and the accuracy as well as reliability of UAV-based vehicle speed models.

LIST OF REFERENCES

Abdallah, M. S., Han, D. S., & Kim, H. W. (2022). Multi-Vehicle Tracking Using Heterogeneous Neural Networks for Appearance And Motion Features. *International Journal of Intelligent Transportation Systems Research*, *20*(3), 720–733. https://doi.org/10.1007/S13177-022-00320-6/FIGURES/11

Abdullah, R. M. (2024). A deep learning-based framework for efficient and accurate 3D real-scene reconstruction. *International Journal of Information Technology (Singapore)*, *16*(7), 4605–4609. https://doi.org/10.1007/S41870-024-02066-8/TABLES/2

Aharchi, M., & Ait Kbir, M. (2020). A Review on 3D Reconstruction Techniques from 2D Images. *Lecture Notes in Intelligent Transportation and Infrastructure*, *Part F1409*, 510–522. https://doi.org/10.1007/978-3-030-37629-1_37/TABLES/2

Andle, J., Soucy, N., Socolow, S., & Sekeh, S. Y. (2023). The Stanford Drone Dataset Is More Complex Than We Think: An Analysis of Key Characteristics. *IEEE Transactions on Intelligent Vehicles*, *8*(2), 1863–1873. https://doi.org/10.1109/TIV.2022.3166642

Arya, D., Maeda, H., Ghosh, S. K., Toshniwal, D., & Sekimoto, Y. (2022). *RDD2022: A multi-national image dataset for automatic Road Damage Detection*. https://arxiv.org/abs/2209.08538v1

Bai, X., Luo, Y., Jiang, L., Gupta, A., Kaveti, P., Singh, H., & Ostadabbas, S. (2024). Bridging the Domain Gap between Synthetic and Real-World Data for Autonomous Driving. *ACM Journal on Autonomous Transportation Systems*, *1*(2), 1–15. https://doi.org/10.1145/3633463

Balamuralidhar, N., Tilon, S., & Nex, F. (2021). MultEYE: Monitoring System for Real-Time Vehicle Detection, Tracking and Speed Estimation from UAV Imagery on Edge-Computing Platforms. *Remote Sensing 2021, Vol. 13, Page 573*, *13*(4), 573. https://doi.org/10.3390/RS13040573

Barisic, A., Petric, F., & Bogdan, S. (2022). Sim2Air - Synthetic Aerial Dataset for UAV Monitoring. *IEEE Robotics and Automation Letters*, *7*(2), 3757–3764. https://doi.org/10.1109/LRA.2022.3147337

Bin Zuraimi, M. A., & Kamaru Zaman, F. H. (2021). Vehicle detection and tracking using YOLO and DeepSORT. *ISCAIE 2021 - IEEE 11th Symposium on Computer Applications and Industrial Electronics*, 23–29. https://doi.org/10.1109/ISCAIE51753.2021.9431784

Borkman, S., Crespi, A., Dhakad, S., Ganguly, S., Hogins, J., Jhang, Y.-C., Kamalzadeh, M., Li, B., Leal, S., Parisi, P., Romero, C., Smith, W., Thaman, A., Warren, S., & Yadav, N. (2021). *Unity Perception: Generate Synthetic Data for Computer Vision*. https://arxiv.org/abs/2107.04259v2

Burdorf, S., Plum, K., & Hasenklever, D. (2022). *Reducing the Amount of Real World Data for Object Detector Training with Synthetic Data*. https://arxiv.org/abs/2202.00632v1

Carranza-García, M., Torres-Mateo, J., Lara-Benítez, P., & García-Gutiérrez, J. (2020). On the Performance of One-Stage and Two-Stage Object Detectors in Autonomous Vehicles Using Camera Data. *Remote Sensing 2021, Vol. 13, Page 89*, *13*(1), 89. https://doi.org/10.3390/RS13010089

Chua, S. Y., Wang, X., Guo, N., Tan, C. S., Chai, T. Y., & Seet, G. L. (2016). Improving three-dimensional (3D) range gated reconstruction through time-of-flight (TOF) imaging analysis. *Journal of the European Optical Society*, *11*. https://doi.org/10.2971/JEOS.2016.16015

Cui, G., Wang, S., Wang, Y., Liu, Z., Yuan, Y., & Wang, Q. (2019). Preceding Vehicle Detection Using Faster R-CNN Based on Speed Classification Random Anchor and Q-Square Penalty Coefficient. *Electronics 2019, Vol. 8, Page 1024*, *8*(9), 1024. https://doi.org/10.3390/ELECTRONICS8091024

Dahl, M., & Javadi, S. (2019). Analytical Modeling for a Video-Based Vehicle Speed Measurement Framework. *Sensors 2020, Vol. 20, Page 160*, *20*(1), 160. https://doi.org/10.3390/S20010160

Damian, A., Filip, C., Nistor, A., Petrariu, I., Mariuc, C., & Stratan, V. (2023). Experimental Results on Synthetic Data Generation in Unreal Engine 5 for Real-World Object Detection. *2023 17th International Conference on Engineering of Modern Electric Systems, EMES 2023*. https://doi.org/10.1109/EMES58375.2023.10171761

Dardagan, N., Brdanin, A., Dzigal, D., & Akagic, A. (2021). Multiple Object Trackers in OpenCV: A Benchmark. *IEEE International Symposium on Industrial Electronics*, *2021-June*. https://doi.org/10.1109/ISIE45552.2021.9576367

Dawn, S., & Biswas, P. (2019). Technologies and methods for 3D reconstruction in archaeology. *Communications in Computer and Information Science*, *968*, 443–453. https://doi.org/10.1007/978-981-13-5758-9_38/FIGURES/6

Dhatbale, R., Bhargava, ·, & Chilukuri, R. (2021). Deep Learning Techniques for Vehicle Trajectory Extraction in Mixed Traffic. *Journal of Big Data Analytics in Transportation 2021 3:2*, *3*(2), 141–157. https://doi.org/10.1007/S42421-021-00042-3

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., & Koltun, V. (2017). *CARLA: An Open Urban Driving Simulator*. https://arxiv.org/pdf/1711.03938

Eilertsen, G., Tsirikoglou, A., Lundström, C., & Unger, J. (2021). *Ensembles of GANs for synthetic training data generation*. https://arxiv.org/abs/2104.11797v1

EUR-Lex - 02019R0947-20220404 - EN - EUR-Lex. (n.d.). Retrieved October 13, 2023, from https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A02019R0947-20220404

Figueira, A., & Vaz, B. (2022). Survey on Synthetic Data Generation, Evaluation Methods and GANs. *Mathematics 2022, Vol. 10, Page 2733*, *10*(15), 2733. https://doi.org/10.3390/MATH10152733

Firdaus, M., & Rau, J. (2017). *Comparisons of the three-dimensional model reconstructed using MicMac, PIX4D mapper and Photoscan Pro*.

Fonder, M., & Van Droogenbroeck, M. (2019). Mid-air: A multi-modal dataset for extremely low altitude drone flights. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, *2019-June*, 553–562. https://doi.org/10.1109/CVPRW.2019.00081

Furukawa, Y., & Ponce, J. (2010). Accurate, dense, and robust multiview stereopsis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *32*(8), 1362–1376. https://doi.org/10.1109/TPAMI.2009.161

Geng, J. (2011). Structured-light 3D surface imaging: a tutorial. *Advances in Optics and Photonics*, *3*(2), 128. https://doi.org/10.1364/aop.3.000128

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2013). Rich feature hierarchies for accurate object detection and semantic segmentation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 580–587. https://doi.org/10.1109/CVPR.2014.81

Glasbey, C. A., van der Heijden, G., Toh, V. F. K., & Gray, A. (2007). Colour displays for categorical images. *Color Research & Application*, *32*(4), 304–309. https://doi.org/10.1002/COL.20327

Gunawan, A. A. S., Tanjung, D. A., & Gunawan, F. E. (2019). Detection of Vehicle Position and Speed using Camera Calibration and Image Projection Methods. *Procedia Computer Science*, *157*, 255–265. https://doi.org/10.1016/J.PROCS.2019.08.165

Haala, N., Rothermel, M., & Cavegn, S. (2015). Extracting 3D urban models from oblique aerial images. *2015 Joint Urban Remote Sensing Event, JURSE 2015*. https://doi.org/10.1109/JURSE.2015.7120479

Huang, J., Artemov, A., Chen, Y., Zhi, S., Xu, K., & Nießner, M. (2023). *SSR-2D: Semantic 3D Scene Reconstruction from 2D Images*. http://arxiv.org/abs/2302.03640

Huang, T. (2018). *Traffic Speed Estimation From Surveillance Video Data* (pp. 161–165).

Huo, W., Yan, Y., Zhou, M., & Li, T. (2022). An improved kernel correlation filter for complex scenes target tracking. *Multimedia Tools and Applications*, *81*(15), 20917–20944. https://doi.org/10.1007/S11042-022-12669-7/FIGURES/16

Ibrahim, ˙, & Deliba¸so˘, D. (2021). *UAV Images Dataset for Moving Object Detection from Moving Cameras*. https://arxiv.org/pdf/2103.11460

Javadi, S., Dahl, M., & Pettersson, M. I. (2019). Vehicle speed measurement model for video-based systems. *Computers & Electrical Engineering*, *76*, 238–248. https://doi.org/10.1016/J.COMPELECENG.2019.04.001

Jiang, J., Mi, C., Wu, M., Zhang, Z., & Feng, Y. (2019). Study on a Real-Time Vehicle Speed Measuring Method at Highway Toll Station. *2019 International Conference on Sensing and Instrumentation in IoT Era, ISSI 2019*. https://doi.org/10.1109/ISSI47111.2019.9043732

Kerbl, B., Kopanas, G., Leimkuehler, T., & Drettakis, G. (2023). 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics*, *42*(4). https://doi.org/10.1145/3592433

Khamis, S., Fanello, S., Rhemann, C., Kowdle, A., Valentin, J., & Izadi, S. (2018). *StereoNet: Guided Hierarchical Refinement for Real-Time Edge-Aware Depth Prediction* (pp. 573–590).

Kim, J. H., Oh, W. T., Choi, J. H., & Park, J. C. (2018). Reliability verification of vehicle speed estimate method in forensic videos. *Forensic Science International*, *287*, 195–206. https://doi.org/10.1016/J.FORSCIINT.2018.04.002

Koutalakis, P., Gkiatas, G., Xinogalos, M., Iakovoglou, V., Kasapidis, I., Pagonis, G., Savvopoulou, A., Krikopoulos, K., Klepousniotis, T., & Zaimes, G. N. (2024). Estimating Stream Bank and Bed Erosion and Deposition with Innovative and Traditional Methods. *Land*, *13*(2). https://doi.org/10.3390/LAND13020232

Lapointe, J. F., Allili, M. S., Belliveau, L., Hebbache, L., Amirkhani, D., & Sekkati, H. (2022). AI-AR for Bridge Inspection by Drone. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *13318 LNCS*, 302–313. https://doi.org/10.1007/978-3-031-06015-1_21/FIGURES/5

Li, J., Chen, S., Zhang, F., Li, E., Yang, T., & Lu, Z. (2019). An Adaptive Framework for Multi-Vehicle Ground Speed Estimation in Airborne Videos. *Remote Sensing 2019, Vol. 11, Page 1241*, *11*(10), 1241. https://doi.org/10.3390/RS11101241

Li, K., Kumar, S., Singh, S. K., Varshney, S., Singh, S., Kumar, P., Kim, B.-G., & Ra, I.-H. (2023). Fusion of Deep Sort and Yolov5 for Effective Vehicle Detection and Tracking Scheme in Real-Time Traffic Management Sustainable System. *Sustainability 2023, Vol. 15, Page 16869*, *15*(24), 16869. https://doi.org/10.3390/SU152416869

Lin, W., Li, Y., Xiao, H., See, J., Zou, J., Xiong, H., Wang, J., & Mei, T. (2019). Group Re-Identification with Multi-grained Matching and Integration. *IEEE Transactions on Cybernetics*, *51*(3), 1478–1492. https://doi.org/10.1109/TCYB.2019.2917713

Little, J. J., & Verri, A. (1989). *Analysis of differential and matching methods for optical flow*. 173–180. https://doi.org/10.1109/WVM.1989.47107

Llorca, D. F., Martínez, A. H., & Daza, I. G. (2021). Vision-based Vehicle Speed Estimation: A Survey. *IET Intelligent Transport Systems*, *15*(8), 987–1005. https://doi.org/10.1049/itr2.12079

Lu, L., & Dai, F. (2024). Accurate road user localization in aerial images captured by unmanned aerial vehicles. *Automation in Construction*, *158*, 105257. https://doi.org/10.1016/J.AUTCON.2023.105257

Lu, Y., Shen, M., Wang, H., Wang, X., van Rechem, C., Fu, T., & Wei, W. (2023). *Machine Learning for Synthetic Data Generation: A Review*. http://arxiv.org/abs/2302.04062

Luo, H., Zhang, J., Liu, X., Zhang, L., & Liu, J. (2024). Large-Scale 3D Reconstruction from Multi-View Imagery: A Comprehensive Review. *Remote Sensing 2024, Vol. 16, Page 773*, *16*(5), 773. https://doi.org/10.3390/RS16050773

Luo, W., Xing, J., Milan, A., Zhang, X., Liu, W., & Kim, T. K. (2021). Multiple object tracking: A literature review. *Artificial Intelligence*, *293*, 103448. https://doi.org/10.1016/J.ARTINT.2020.103448

Lyssenko, M., Gladisch, C., Heinzemann, C., Woehrle, M., & Triebel, R. (n.d.). *Instance Segmentation in CARLA: Methodology and Analysis for Pedestrian-oriented Synthetic Data Generation in Crowded Scenes*. Retrieved October 25, 2024, from https://github.com/carla-simulator/carla/

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., & Ng, R. (2020). NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *12346 LNCS*, 405–421. https://doi.org/10.1007/978-3-030-58452-8_24

Musgrove, C., Naething, R., Schilling Cameron Musgrove, J., & Schilling, J. (2014). Arbitrary scene simulation for synthetic aperture radar. *Https://Doi.Org/10.1117/12.2049875*, *9077*, 65–74. https://doi.org/10.1117/12.2049875

Nikolenko, S. I. (2019). Synthetic Data for Deep Learning. *Springer Optimization and Its Applications*, *174*, 1–54. https://doi.org/10.1007/978-3-030-75178-4_1

Nowruzi, F. E., Kapoor, P., Kolhatkar, D., Hassanat, F. Al, Laganiere, R., & Rebut, J. (2019). *How much real data do we actually need: Analyzing object detection performance using synthetic and real data*. https://arxiv.org/pdf/1907.07061

Özcan, İ., Altun, Y., & Parlak, C. (2024). Improving YOLO Detection Performance of Autonomous Vehicles in Adverse Weather Conditions Using Metaheuristic Algorithms. *Applied Sciences 2024, Vol. 14, Page 5841*, *14*(13), 5841. https://doi.org/10.3390/APP14135841

Pan, X., Dai, B., Liu, Z., Loy, C. C., & Luo, P. (2020). Do 2D GANs Know 3D Shape? Unsupervised 3D shape reconstruction from 2D Image GANs. *ICLR 2021 - 9th International Conference on Learning Representations*. https://arxiv.org/abs/2011.00844v4

Pepe, M., Fregonese, L., & Crocetto, N. (2022). Use of SfM-MVS approach to nadir and oblique images generated throught aerial cameras to build 2.5D map and 3D models in urban areas. *Geocarto International*, *37*(1), 120–141. https://doi.org/10.1080/10106049.2019.1700558

Pollok, T., Junglas, L., Ruf, B., & Schumann, A. (2019). UnrealGT: Using Unreal Engine to Generate Ground Truth Datasets. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *11844 LNCS*, 670–682. https://doi.org/10.1007/978-3-030-33720-9_52/FIGURES/9

Pomerleau, D. A. (1989). *ALVINN: AN AUTONOMOUS LAND VEHICLE IN A NEURAL NETWORK*.

Rahmani, S., Rieder, S., de Gelder, E., Sonntag, M., Mallada, J. L., Kalisvaart, S., Hashemi, V., & Calvert, S. C. (2024). *A Systematic Review of Edge Case Detection in Automated Driving: Methods, Challenges and Future Directions*. https://arxiv.org/pdf/2410.08491v1

Raistrick, A., Lipson, L., Ma, Z., Mei, L., Wang, M., Zuo, Y., Kayan, K., Wen, H., Han, B., Wang, Y., Newell, A., Law, H., Goyal, A., Yang, K., & Deng, J. (2023). *Infinite Photorealistic Worlds using Procedural Generation*. 12630–12641. https://doi.org/10.1109/cvpr52729.2023.01215

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2016-December*, 779–788. https://doi.org/10.1109/CVPR.2016.91

Reis, D., Kupec, J., Hong, J., & Daoudi, A. (2023). *Real-Time Flying Object Detection with YOLOv8*. https://arxiv.org/abs/2305.09972v1

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(6), 1137–1149. https://doi.org/10.1109/TPAMI.2016.2577031

Samavati, T., & Soryani, M. (2023). Deep learning-based 3D reconstruction: a survey. *Artificial Intelligence Review*, *56*(9), 9175–9219. https://doi.org/10.1007/S10462-023-10399-2/FIGURES/12

Shah, S., Dey, D., Lovett, C., & Kapoor, A. (2017). AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles. *Springer Proceedings in Advanced Robotics*, *5*, 621–635. https://doi.org/10.1007/978-3-319-67361-5_40

Shao, Y., Yang, Z., Li, Z., & Li, J. (2024). Aero-YOLO: An Efficient Vehicle and Pedestrian Detection Algorithm Based on Unmanned Aerial Imagery. *Electronics 2024, Vol. 13, Page 1190*, *13*(7), 1190. https://doi.org/10.3390/ELECTRONICS13071190

Shaqib, S., Alo, A. P., Ramit, S. S., Rupak, A. U. H., Khan, S. S., & Rahman, M. S. (2024). *Vehicle Speed Detection System Utilizing YOLOv8: Enhancing Road Safety and Traffic Management for Metropolitan Areas*. https://arxiv.org/pdf/2406.07710v1

Silva, L. A., Leithardt, V. R. Q., Batista, V. F. L., Villarrubia Gonzalez, G., & De Paz Santana, J. F. (2023). Automated Road Damage Detection Using UAV Images and Deep Learning Techniques. *IEEE Access*, *11*, 62918–62931. https://doi.org/10.1109/ACCESS.2023.3287770

Sohan, M., Ram, T. S., Venkata, C., & Reddy, R. (2024). *A Review on YOLOv8 and Its Advancements*. 529–545. https://doi.org/10.1007/978-981-99-7962-2_39

Srivastava, A., & Prakash, J. (2023). Techniques, Answers, and Real-World UAV Implementations for Precision Farming. *Wireless Personal Communications*, *131*(4), 2715–2746. https://doi.org/10.1007/S11277-023-10577-Z/FIGURES/11

Tang, H., & Jia, K. (2023). A New Benchmark: On the Utility of Synthetic Data with Blender for Bare Supervised Learning and Downstream Domain Adaptation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2023-June*, 15954–15964. https://doi.org/10.1109/CVPR52729.2023.01531

Terven, J. R., & Cordova-Esparaza, D. M. (2023). *A COMPREHENSIVE REVIEW OF YOLO: FROM YOLOV1 TO YOLOV8 AND BEYOND UNDER REVIEW IN ACM COMPUTING SURVEYS*.

Terven, J. R., & Cordova-Esparza, D. M. (2023). *A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS*. https://doi.org/10.3390/make5040083

Thompson, S., Celebi, M. E., & Buck, K. H. (2020). Fast color quantization using MacQueen's k-means algorithm. *Journal of Real-Time Image Processing*, *17*(5), 1609–1624. https://doi.org/10.1007/S11554-019-00914-6/TABLES/4

Tilon, S. M., & Nex, F. (n.d.). *VEHICLE TRACKING AND SPEED ESTIMATION FROM UNMANNED AERIAL VEHICLES USING SEGMENTATION-INITIALISED TRACKERS*. https://doi.org/10.5194/isprs-annals-X-1-W1-2023-431-2023

Torres-Sánchez, J., López-Granados, F., Borra-Serrano, I., & Manuel Peña, J. (2018). Assessing UAV-collected image overlap influence on computation time and digital surface model accuracy in olive orchards. *Precision Agriculture*, *19*(1), 115–133. https://doi.org/10.1007/S11119-017-9502-0/FIGURES/10

Tyagi, B., Nigam, S., & Singh, R. (2023). Human Detection and Tracking Based on YOLOv3 and DeepSORT. *Lecture Notes in Networks and Systems*, *686 LNNS*, 125–135. https://doi.org/10.1007/978-981-99-2100-3_11/FIGURES/7

Vakili, E., Shoaran, M., & Sarmadi, M. R. (2020). Single–camera vehicle speed measurement using the geometry of the imaging system. *Multimedia Tools and Applications*, *79*(27–28), 19307–19327. https://doi.org/10.1007/S11042-020-08761-5/FIGURES/15

Waar mag ik niet vliegen met een drone? | Rijksoverheid.nl. (n.d.). Retrieved October 13, 2023, from https://www.rijksoverheid.nl/onderwerpen/drone/vraag-en-antwoord/waar-mag-ik-niet-vliegen-met-een-drone

Westoby, M. J., Brasington, J., Glasser, N. F., Hambrey, M. J., & Reynolds, J. M. (2012). 'Structure-from-Motion' photogrammetry: A low-cost, effective tool for geoscience applications. *Geomorphology*, *179*, 300–314. https://doi.org/10.1016/J.GEOMORPH.2012.08.021

Wojke, N., Bewley, A., & Paulus, D. (2018). Simple online and realtime tracking with a deep association metric. *Proceedings - International Conference on Image Processing, ICIP*, *2017-September*, 3645–3649. https://doi.org/10.1109/ICIP.2017.8296962

Wu, X., Li, W., Hong, D., Tao, R., & Du, Q. (2022). Deep Learning for Unmanned Aerial Vehicle-Based Object Detection and Tracking: A survey. *IEEE Geoscience and Remote Sensing Magazine*, *10*(1), 91–124. https://doi.org/10.1109/MGRS.2021.3115137

Xing, D., & Tzes, A. (2023). Synthetic Aerial Dataset for UAV Detection via Text-to-Image Diffusion Models. *Proceedings - 2023 IEEE Conference on Artificial Intelligence, CAI 2023*, 51–52. https://doi.org/10.1109/CAI54212.2023.00030

Xiong, B., Li, Z., & Li, Z. (2024). *GauU-Scene: A Scene Reconstruction Benchmark on Large Scale 3D Reconstruction Dataset Using Gaussian Splatting*. https://arxiv.org/abs/2401.14032v1

Yang, B., Tang, M., Chen, S., Wang, G., Tan, Y., & Li, B. (2020). A vehicle tracking algorithm combining detector and tracker. *Eurasip Journal on Image and Video Processing*, *2020*(1), 1–20. https://doi.org/10.1186/S13640-020-00505-7/TABLES/1

Yang, L., Li, M., Song, X., Xiong, Z., Hou, C., & Qu, B. (2019). Vehicle Speed Measurement Based on Binocular Stereovision System. *IEEE Access*, *7*, 106628–106641. https://doi.org/10.1109/ACCESS.2019.2932120

Yin, G., Yu, M., Wang, M., Hu, Y., & Zhang, Y. (2022). Research on highway vehicle detection based on faster R-CNN and domain adaptation. *Applied Intelligence*, *52*(4), 3483–3498. https://doi.org/10.1007/S10489-021-02552-7/TABLES/6

Zarindast, A., & Sharma, · Anuj. (2023). Opportunities and Challenges in Vehicle Tracking: A Computer Vision-Based Vehicle Tracking System. *Data Science for Transportation 2023 5:1*, *5*(1), 1–12. https://doi.org/10.1007/S42421-023-00063-0

Zhang, Y., Sun, P., Jiang, Y., Yu, D., Weng, F., Yuan, Z., Luo, P., Liu, W., & Wang, X. (2022). ByteTrack: Multi-object Tracking by Associating Every Detection Box. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *13682 LNCS*, 1–21. https://doi.org/10.1007/978-3-031-20047-2_1/TABLES/7

Zhang, Z., He, T., Zhang, H., Zhang, Z., Xie, J., & Li, M. (2019). *Bag of Freebies for Training Object Detection Neural Networks*. https://arxiv.org/pdf/1902.04103

Zhu, P., Wen, L., Du, D., Bian, X., Fan, H., Hu, Q., & Ling, H. (2022). Detection and Tracking Meet Drones Challenge. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *44*(11), 7380–7399. https://doi.org/10.1109/TPAMI.2021.3119563

Zhu, Q., Min, C., Wei, Z., Chen, Y., & Wang, G. (2021). *Deep Learning for Multi-View Stereo via Plane Sweep: A Survey*. http://arxiv.org/abs/2106.15328