

TDK-thesis

Nickodemus Richard Rinaldi

Relative Pose Estimation of Car-Mounted Camera Using Drone-Mounted Camera's Image

EÖTVÖS LORÁND UNIVERSITY

FACULTY OF INFORMATICS

DEPARTMENT OF ALGORITHMS AND THEIR APPLICATIONS



Author:

Nickodemus Richard Rinaldi

Computer Science MSc

V. grade

Supervisor:

Dr. Hajder Levente

Research Fellow

Budapest, 2024

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Related Work	2
1.3	Goal	6
1.4	Contribution	7
2	Specification and Design	8
2.1	CARLA Environment	8
2.2	Vehicle	11
2.3	Simulation Setup and Dataset Generation	12
2.4	Real World Experiment	15
2.4.1	Experiment One	17
2.4.2	Experiment Two	18
3	Implementation and Result	21
3.1	Theoretical Background	21
3.1.1	Camera Projection	21
3.1.2	Pose Estimation	24
3.1.3	Homography	25
3.2	Algorithm Development	26
3.2.1	Pose Estimation on Each Camera	28
3.2.2	Concatenating Pose Between Cameras	31
3.2.3	Chessboard Detection	34
3.2.4	Vehicle's Camera Pose from The Chessboard	37
3.2.5	Relative Pose Estimation of Car-Mounted Camera from Drone-Mounted Camera	37
3.2.6	Homography Matrix Composition	38
3.3	Result and Discussion	42

4 Conclusion and Future Works	48
Acknowledgements	50
A Perspective-n-Points	51
A.1 Linear Algorithms	52
A.2 Iterative Non-Linear Algorithms	52
A.3 Non-Iterative Non-Linear Algorithm	53
B Homography from Motion	54
Bibliography	56
List of Figures	61
List of Algorithms	64

Abstract

Knowing the traffic information around the car is essential in developing the self-driving car. The experiment is conducted to determine the possibility of using a drone camera. The drone-mounted camera can be used to capture the image of the car from above, widely known as a bird's-eye-view (BEV) image. BEV image could give more information about the car's surrounding environment. This work simulated the usage of a drone-mounted camera in CARLA, an autonomous driving car simulator. An algorithm was developed that could transform the image captured by a drone into an image captured by a car-mounted camera. First, both cameras were calibrated, and then a common object was used to determine the transformation between the drone-mounted camera and the car-mounted camera. A chessboard was installed on top of the car as a reference for the drone-mounted camera to follow the vehicle. Pose estimation from the chessboard to the car-mounted camera was then used to estimate the homography between the drone and the car-mounted camera as the car moved. The algorithm has also been successfully tested in real-world data captured using ELTE-car.

Chapter 1

Introduction

1.1 Motivation

A critical aspect of developing autonomous driving is getting good information about the vehicle's surroundings. Several methods have been developed, including combining different sensors, known as sensor fusion. Typical sensors used in autonomous driving are digital cameras, LiDAR, radar, IMU, GNSS, depth cameras, etc. Combining several sensors gives more information about the vehicle's surroundings. Some post-processing of the data is also carried out to have a better sense of the environment. One of these post-processing is bird-eye-view (BEV) transformation [1]. BEV gives a better understanding of the traffic situation in the surrounding area. Autonomous driving research using BEV is gaining interest in some generations of BEV datasets [2][3].

In addition to data processing, developing the data acquisition process is equally important. This work explored the usage of digital camera sensors for data collection. The BEV will be captured using a drone-mounted camera that follows the car's movement. Afterwards, the images taken using a drone-mounted camera will be combined with the car-mounted image. Before combining images taken from a car-mounted and a drone-mounted camera, both cameras need to be calibrated between each other. The calibration method proposed by Zhang et al. [4] was used for calibration. The pose estimation [5] of each camera was computed by solving the pose estimation problem using a perspective- n -point (PnP) algorithm [6][7][8][9].

Conducting direct on-road tests to prove the concept poses significant challenges. The high cost and regulatory issues of flying drones in open streets necessitate a thorough feasibility and effectiveness test. To this end, concept proofing will be conducted in a

simulation environment that can replicate real-life traffic scenarios. This will provide a clear understanding of the method’s practicality and performance.

Several simulation environments support the research of autonomous driving [10]. Among them, the open-source autonomous driving simulator CARLA [11] is widely used because of its ability to simulate urban traffic situations and its array of built-in sensors relevant to autonomous driving. It also renders an excellent quality visualisation. Given these advantages, CARLA has been chosen as the simulation environment for this study.

1.2 Related Work

In the work of Krajewski et al. [2], the drone hovered above German highways. The recording took place in sunny and windless weather. The captured videos were stabilised using OpenCV by mapping each frame’s background with the first frame’s background image. Post-processing was done to annotate the video, which given the result in Figure 1.1

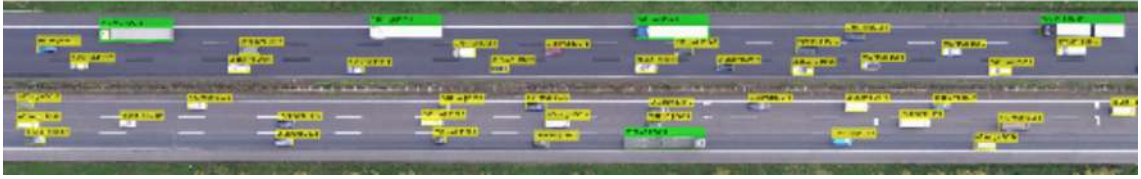


Figure 1.1: Example of the highD dataset [2]

Yang et al. [3] created two datasets of a pedestrian trajectory called CITR and DUT. The CITR dataset was generated in a controlled environment in a parking lot of the Control and Intelligent Transportation Research (CITR) Lab at The Ohio State University (OSU). A golf cart with a marker on top of it to track the motion drove through a parking lot where some pedestrians were crossing. The vehicle agent (a golf car) shown in Figure 1.2 moved back and forth in a designated space following the scenario in Figure 1.3. At the same time, the drone stayed statically in the sky to record the situation.



Figure 1.2: EZ Golf Cart as vehicle agent (left) and the markers on top of the car (right).[3]

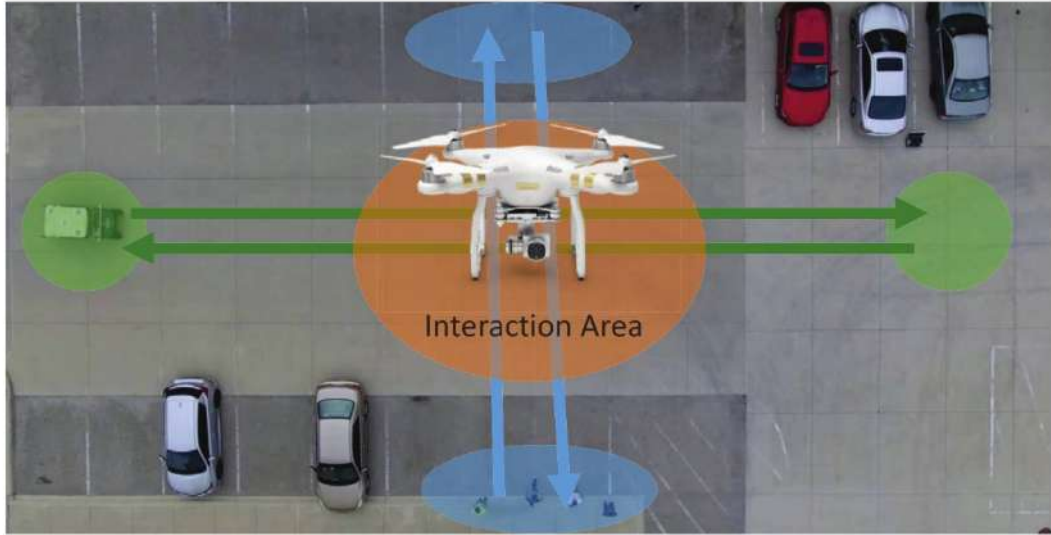


Figure 1.3: CITR Dataset Setup: the pedestrians moved between two blue areas, the car moved between two green areas, and the drone stayed in the orange area.[3]

The DUT dataset was captured in a crowded area of China's Dalian University of Technology (DUT) campus. The videos were taken in two places, one on a road intersection with a pedestrian crosswalk and the other on a roundabout. As for the CITR data capture, the drone only hovered in the designated position. Data generation of the DUT dataset can be seen in Figure 1.4

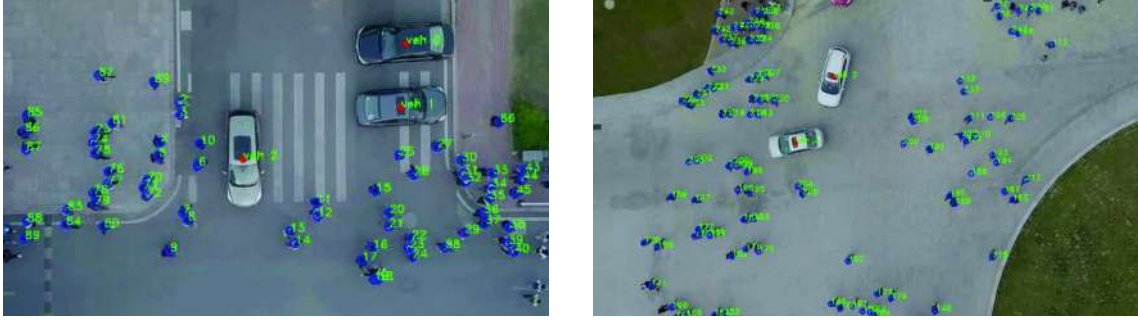


Figure 1.4: left: capturing pedestrian crosswalk, right: capturing a roundabout area.[3]

Robicquet et al. [12] created a large dataset of BEV videos to research the human-vehicle interaction. A quadcopter was used to take videos from more than 100 different crowded spaces at Stanford University, as shown in Figure 1.5. Various objects in the videos, such as pedestrians, bikers, buses, cars, buses, skateboarders, and golf carts, were then annotated, as can be seen in Figure 1.6

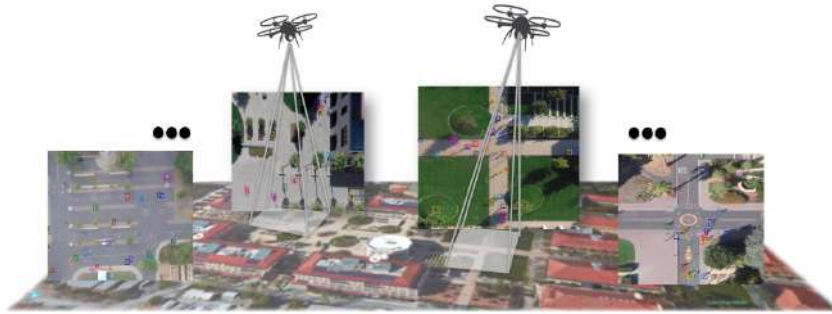


Figure 1.5: Experimental Setup of Human-Social Dataset[12]



Figure 1.6: Human-Social Dataset Example[12]

Another example of BEV datasets is the INTERACTION dataset [13]. The datasets were captured using drones and traffic cameras in different locations worldwide. The datasets aimed to show interactive driving scenarios in various countries. An example of the dataset is shown in Figure 1.7

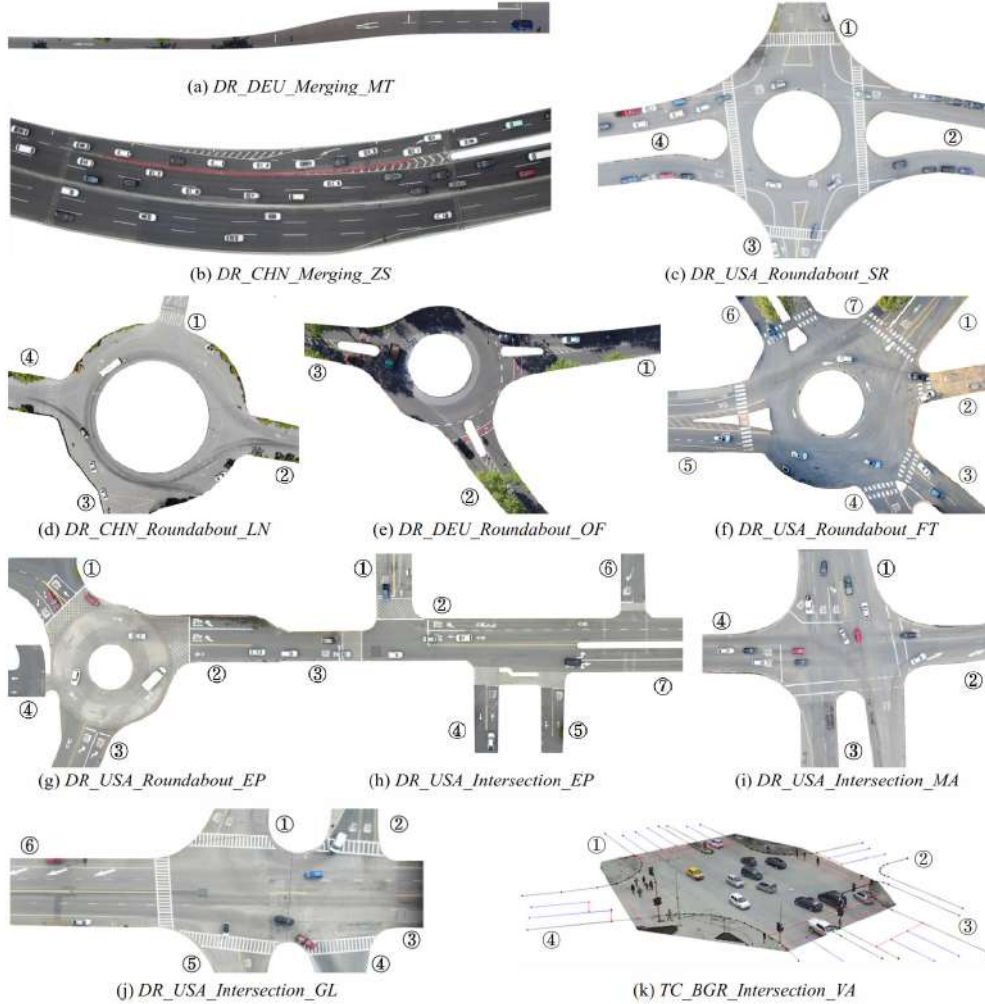


Figure 1.7: INTERACTION datasets example (a) urban merging, (b) highway ramp merging and lane change, (c)-(g) five roundabouts, and (h)-(j) unsignalised intersections, and (k) unprotected left turn at a signalised intersection.[13]

Camera motion and planar equation can be derived if two images of the same planar plane were taken from different angles. It is widely known as a structure from motion case [7][14]. Usually, the same camera was used to take images of the reference planes from different perspectives. Given the camera's intrinsic parameter, the homography between images can be derived.

Projective transformation is a technique used in image processing to transform an image from one perspective to another. One example of this transformation is a regular

image transformed into a bird's eye view (BEV). This technique has been used in various applications, including developing lane detection algorithms [15], as well as transforming images to assist drivers in parking [16]. The BEV provides a top-down view of the scene, which can help detect objects and accurately map the environment.

Augmented reality (AR) is a technology where 3D virtual objects are integrated into the 3D real-time environment [17]. Therefore, pose estimation to derive the position of a camera plays an essential role in this field. Several methods to solve the pose estimation problem have been developed in this field. Some survey papers like [5][18] were available for the readers to dig deeper into AR. Both papers state that the tracking technique is currently the most used method to solve the pose estimation problem. However, some state-of-the-art methods like Simultaneous Localization and Mapping (SLAM) [19][20][21] gain some attentions nowadays. The tracking technique means that a sensor tracks a certain object in the real world and estimates the camera pose by projecting points in the 3D coordinate world system into the 2D image coordinates. A few examples of an object used as a tracker object were a square marker [22] or a chessboard corner [23]. Inspired by this tracker approach, the camera mounted on the drone will track an easily identified object, a chessboard pattern installed on the car's roof. The usage of aerial images for visual tracking was also used in the aerospace area for the research of the development of Unmanned Aerial Vehicle (UAV) [24][25]

1.3 Goal

As to the extent of the writer's knowledge, there was not a published dataset yet of BEV image captured using a drone following the car movement. The resulting BEV image can be transformed into a frontal view image using a homography matrix. The homography can be composed when we know the pose estimation of a car-mounted camera to the drone-mounted camera. This work aimed to write an algorithm to compute the relative pose between the car-mounted and drone-mounted cameras. First, each camera's pose was determined using a PnP algorithm, and the estimated poses were used to determine the relative pose between the car-mounted camera and the drone-mounted camera. The final homography matrix was used to project the BEV from the drone-mounted camera into the frontal view image. Autonomous driving simulation, CARLA [11], was used to model the setup of a drone following a car and record

the images. The algorithm was implemented on the synthetic data and then tested on real-world data captured by the ELTE Car and DJI Drone.

1.4 Contribution

The new method to find the relative pose between cameras with different points of view was a novelty that this work offered as a contribution to computer vision research. The proposed method develops a way of creating image transformation from different perspectives.

For this thesis work, a new car model with a chessboard was modelled in a 3D modeller software and added to the CARLA simulator. The car model can be used as a CARLA asset for developing a motion-tracking algorithm or another research that needs a calibrated pattern attached to a vehicle. The simulation process and real-world experiments tested in this work could be used as a setup example to generate another dataset for aerial image analysis, motion planning, or localisation and mapping, to name some examples of possible usage of the datasets in various research areas.

Chapter 2

Specification and Design

As proposed by Zhang[4], camera calibration was done using a printed pattern attached to a planar surface. A chessboard pattern was used as the calibration pattern. The chessboard was attached to the top of the vehicle. On the other side, an easily recognisable pattern that both car-mounted and drone-mounted cameras can see was used to find the relative pose between both cameras. In CARLA, the pedestrian crosswalk was used. Then, the vehicle ran on several scenarios. Several images were captured in CARLA, and the suggested algorithm was implemented. In the real-world experiment, the markers were put on the road. A detailed explanation of CARLA, the vehicle model, and the synthetic dataset will be explained through Sections 2.1 to 2.3, and Section 2.4 will describe the setting of a real-world experiment conducted using ELTE's Car and DJI drone in the ELTE's parking lot.

2.1 CARLA Environment

Motion planning algorithms are one of the crucial aspects of the development of autonomous vehicles. Therefore, many simulation tools have been developed to test the motion planning algorithm before testing it in real-world scenarios to minimise the cost and prevent accidents during the test. There are a variety of open-source and commercial simulation tools. Tong et al. [10] have created a survey comparing different simulation tools that have been used recently, to name some: CARLA, Gazebo, SUMO, CarMaker, etc. These simulation tools range from applications in robotics and some are specifically for vehicle simulation. An extensive explanation can be read in [10]. Our work needed

sensor simulation and good graphic rendering. Following the suggestion from [10], CARLA was used as a simulator in this work.

CARLA is an open-source autonomous driving simulator[11]. Besides the open-source aspect, CARLA can also simulate the traffic situation clearly, which was also a consideration in our work. The latest release (CARLA 0.9.15) was built on Unreal Engine 4.26. CARLA is based on the architecture of a client-server. The server simulates the world. The client retrieves information and demands changes in the world. The client can connect with the world through the PythonAPI. CARLA can run in asynchronous and synchronous mode. In asynchronous mode, the server runs the simulation as fast as possible without waiting for a call from the client. While in synchronous mode, the server will wait for a client's tick before updating its simulation state.

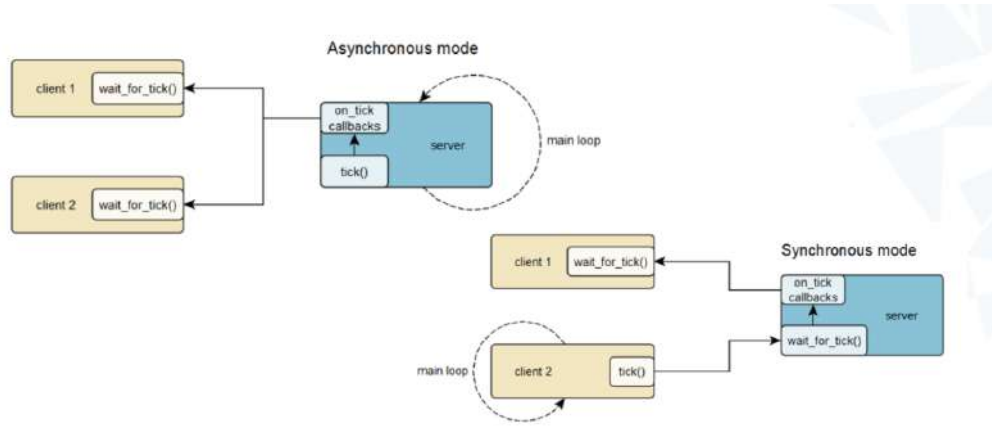


Figure 2.1: Carla Architecture.[11]

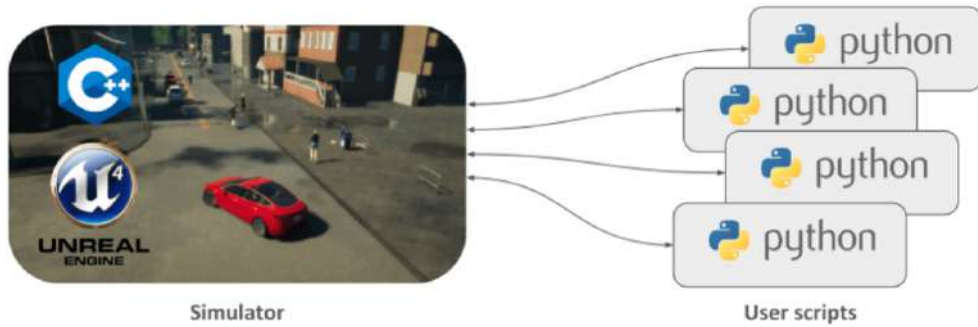


Figure 2.2: Visualization of Carla Modules.[11]

CARLA can be installed using a package installation or built from the source code. Building from the source is needed for development purposes. After building from the source, a user can create additional assets or manipulate maps using Unreal Editor. A

build-from-source method was chosen for this work because the modification of the assets was needed by adding a car with a chessboard attached.

There are already 12 built-in maps in CARLA. The default Town 10 was used in this work. It pictures the urban area with skyscraping buildings and road junctions, which are already enough to carry out our simulation.



Figure 2.3: Top View of CARLA's Town 10 maps



Figure 2.4: Example View of Carla Town 10 Maps

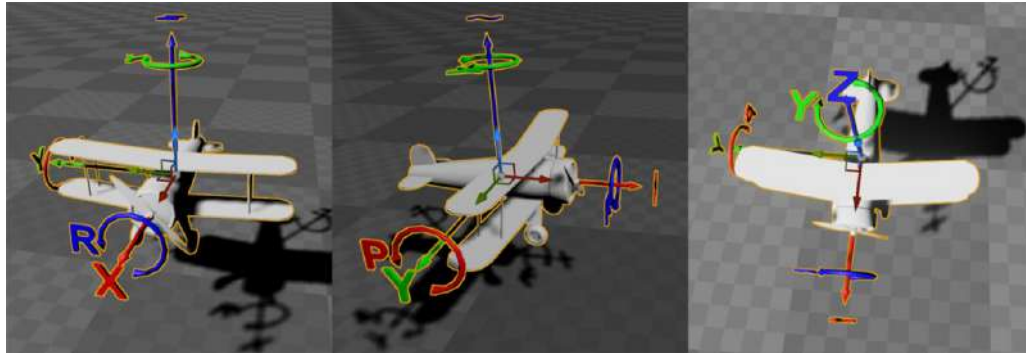


Figure 2.5: CARLA uses the Unreal Engine coordinate system (left-handed system)[11].

2.2 Vehicle

The vehicle was modelled using a 3D modeller software, Blender. A 5 x 4 chessboard-like pattern with a square size of 30 cm was attached to the top of the car as shown in Figure 2.6

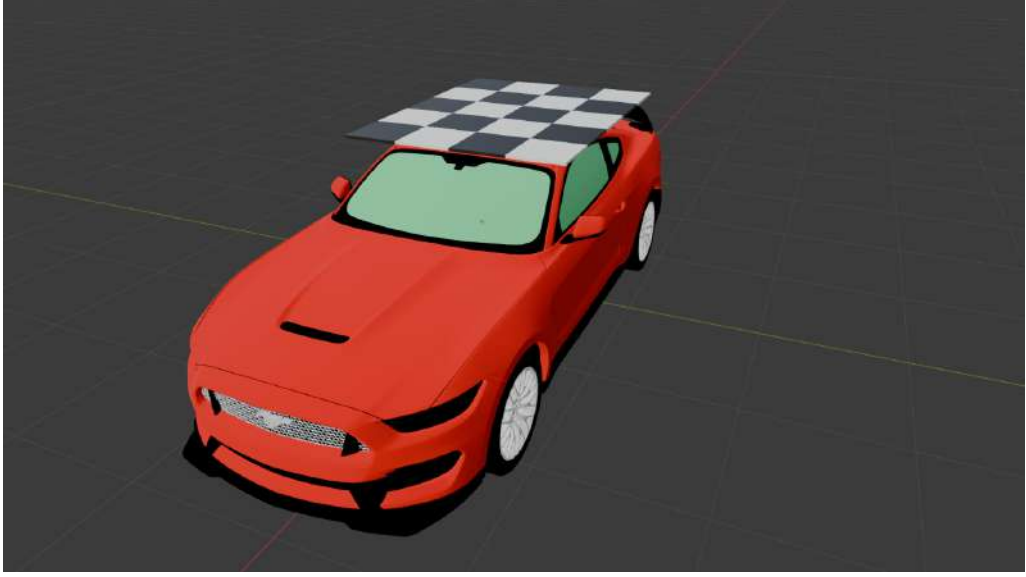


Figure 2.6: Car Model

2.3 Simulation Setup and Dataset Generation

CARLA offers a wide range of sensors commonly used in autonomous driving research, including RGB cameras, RGB-D cameras, LiDAR, GNSS, and IMU. In our work, we utilised two RGB cameras. The first, a front-view camera, was fixed on the vehicle's hood, while the second was positioned in the sky to mimic a drone-mounted camera. We simulated three different heights of the drone above the road: 15 meters, 25 meters, and 35 meters. Furthermore, the drone's movement due to wind or other instability was simulated by adding rotation, translation, and a combination of rotation and translation to the drone-mounted camera.



Figure 2.7: Image taken by car-mounted camera

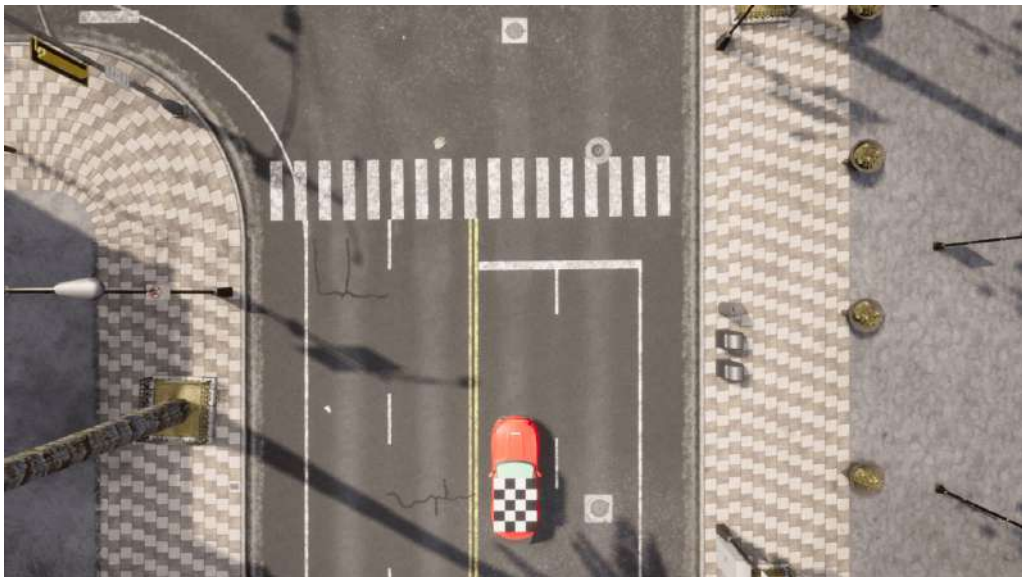


Figure 2.8: Image taken by a 15 meters drone-mounted camera

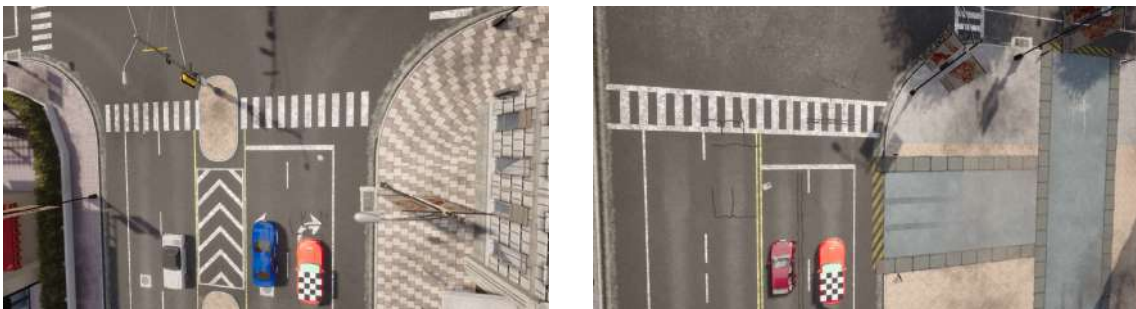


Figure 2.9: Rotation effect on a drone-mounted camera positioned at 15 meters height.



Figure 2.10: Image taken by a 25 meters drone-mounted camera

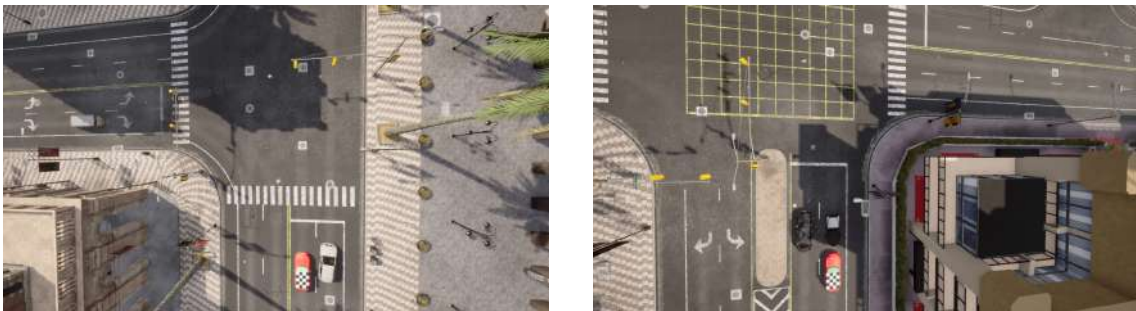


Figure 2.11: Rotation effect on a drone-mounted camera positioned at 25 meters height.



Figure 2.12: Image taken by a 35 meters drone-mounted camera



Figure 2.13: Rotation effect on a drone-mounted camera positioned at 35 meters height.

Around 1-2 minutes length videos with a resolution of 1920 x 1080 pixels were recorded for each scenario. In the CARLA simulator, it is possible to set up the horizontal field-of-view (FOV) and the image size that the RGB camera needs to generate.

The crosswalk seen at the beginning of the frame, like in Figure 2.7 was used as a reference object to calibrate the car-mounted and drone-mounted cameras. The dimensions of the crosswalk were measured in the Unreal Editor.

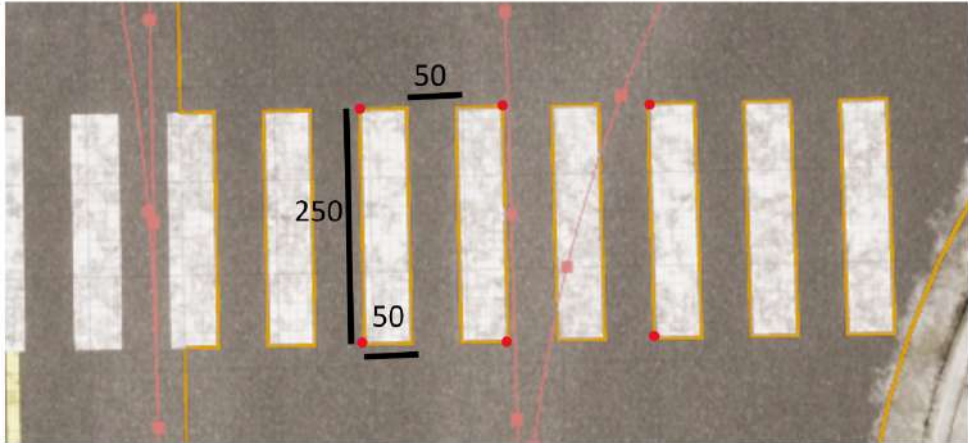


Figure 2.14: CARLA's crosswalk dimension.

2.4 Real World Experiment

The real-world data was also recorded to test the algorithm. The recording was done in ELTE's parking space using the ELTE Car and a drone. Figure 2.15 shows that the chessboard pattern and four calibrated cameras were installed on the roof of the ELTE Car. The camera configuration can be seen in Figure 2.16. There were two standard cameras, marked as Dev0 and Dev3 cameras and two fish-eye lens cameras, marked as Dev1 and

Dev2 cameras. The cameras have a resolution of 1920×1200 px and a pixel size of $4.8 \times 4.8 \mu\text{m}^2$.



Figure 2.15: ELTE Car Camera Installation

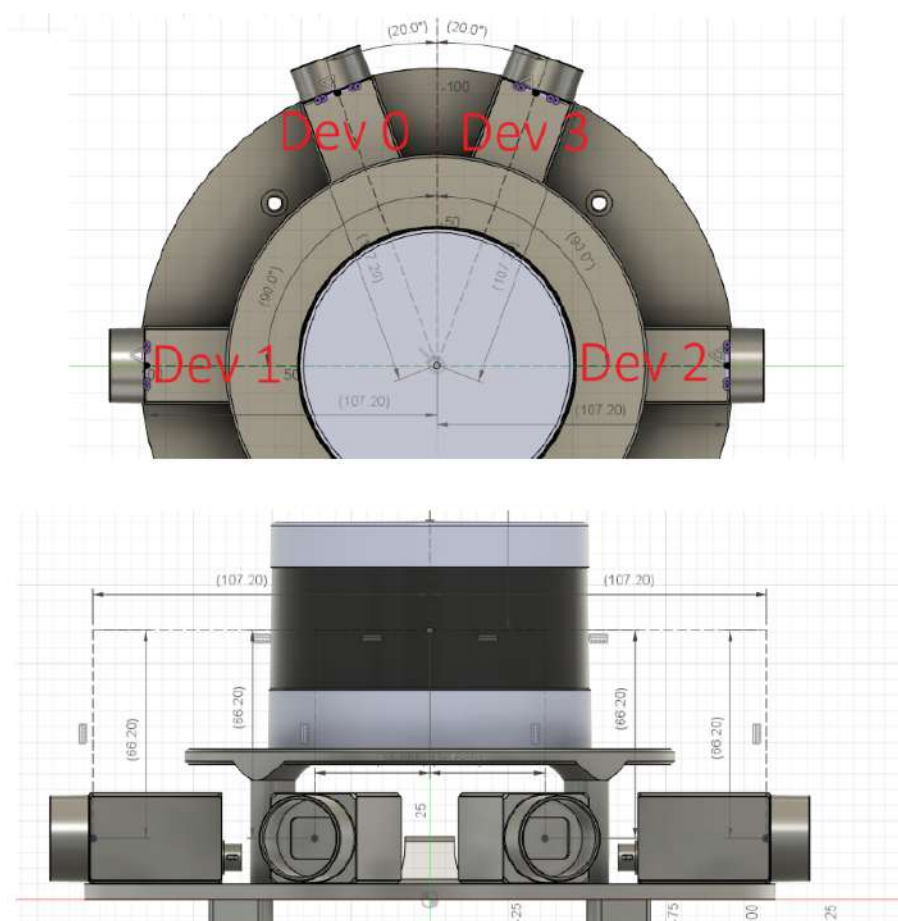


Figure 2.16: ELTE Car Camera Configuration

A pilot controlled a drone to follow the movement of the car. As an alternative to the crosswalk seen in the CARLA simulation, six markers were placed in front of the vehicle for the object reference, as in Figure 2.17 configuration. The distance between each marker is 280 cm long and 340 cm wide. The car-mounted camera and the drone-mounted camera were aligned using these markers. A person gave a signal as a sign for time synchronisation. Two experiments were conducted for this work.



Figure 2.17: Real World Experiment: Markers Position.

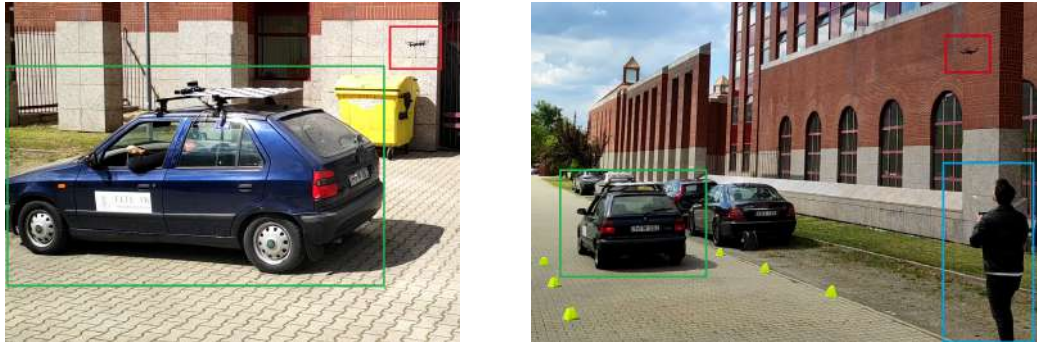


Figure 2.18: ELTE Experiment. Green box: ELTE Car, Red Box: Drone, Blue Box: Drone Pilot

2.4.1 Experiment One

The first experiment used a DJI Tello drone, which only had a front-view camera. This limitation made the drone-mounted camera face the same view as the car-mounted camera, only with different altitudes, as shown in Figure 2.19. Although this view differed from the simulated data and did not represent the BEV image, the experiment could

demonstrate how the proposed algorithm worked in various situations. We used the ELTE's Camera Dev3's image for experiment one as it clearly captured the marker in the first frame, as seen in Figure 2.20, for the alignment of the car-mounted camera and drone-mounted camera.



Figure 2.19: Example Image Captured by Drone-Mounted Camera



(a) Camera Dev0



(b) Camera Dev1



(c) Camera Dev2



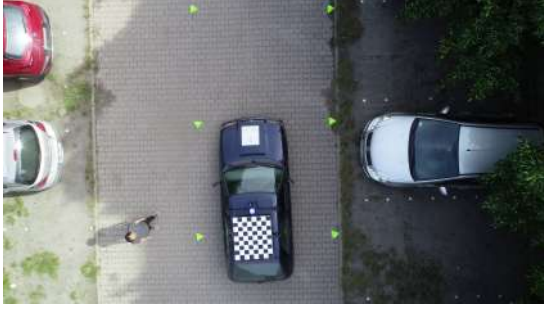
(d) Camera Dev3

Figure 2.20: Example of images taken by ELTE Car in experiment one.

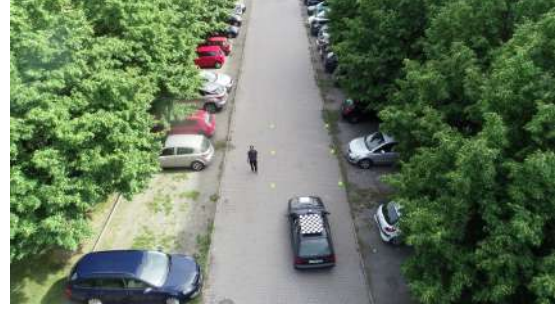
2.4.2 Experiment Two

The second experiment was carried out using DJI Phantom 4 Pro V2.0 drone. The drone has a stabiliser, making it more sturdy and able to capture images better. The drone can fly higher than the DJI Tello drone; according to the manufacturer, it could reach

a maximum service ceiling of up to 6,000 meters above sea level. The drone also has a downward vision system; therefore, it could capture aerial images similar to a BEV image we simulated in CARLA, as shown in Figure 2.21. The video the DJI Phantom 4 drone recorded has a 4K resolution (3840×2160).



(a) 10 meters height.



(b) 15 meters height.



(c) 20 meters height.



(d) 20 meters height.

Figure 2.21: Image Captured by Drone DJI Phantom 4 Pro V2.0

During the experiment, the drone flies at around an altitude of 15 meters, following the ELTE Car. In experiment two, images captured from a car-mounted camera Dev0 were used. Figure 2.22 shows that all the markers were clearly visible from Dev0. Therefore, the images could be used as image pairs with the images from the drone-mounted camera.



(a) Camera Dev0



(b) Camera Dev1



(c) Camera Dev2



(d) Camera Dev3

Figure 2.22: Example of images taken by ELTE Car in experiment two. (Notice a person gave a sign for time synchronisation in Dev0)

Chapter 3

Implementation and Result

In this chapter, we provided the development of the algorithm and its implementation. We briefly explained the theoretical concept related to the algorithm in Section 3.1. We then built upon these concepts a step-by-step explanation of the proposed algorithm, along with some examples, in Section 3.2. Finally, the discussion of the results of the algorithm implementation was presented in Section 3.3.

3.1 Theoretical Background

3.1.1 Camera Projection

In computer vision, a point in a world coordinate system $\begin{bmatrix} x & y & z \end{bmatrix}^T$ can be projected to a point in an image coordinate system $\begin{bmatrix} u & v \end{bmatrix}^T$ using a camera's intrinsic parameter and extrinsic parameters. The intrinsic parameter, known as the camera matrix, $\mathbf{K}$, consists of the camera focal length f_x, f_y , and the principal point or optical centre c_x, c_y . In practice, it was often to set the focal length as the same, $f_x = f_y = f$, and the optical centre set at the centre of the image[26], $(c_x, c_y) = (image_width/2, image_height/2)$. The extrinsic parameters are rotation matrix, $\mathbf{R}$, and translation vector $\mathbf{t}$. The 3D-2D projection of a pinhole camera is described by Equation 3.2[7][26]

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \quad (3.1)$$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K}[\mathbf{R} \mid \mathbf{t}] \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (3.2)$$

Due to lens distortion, sensor misalignment, environmental factors, etc., there can be distortion in the image. Therefore, in computer vision applications, it is essential to do a camera calibration[27][28][29]. There were many camera calibration methods; some examples were using epipoles line between two images called self-calibration suggested by Faugeras et al.[30], a four-step procedure by Heikkilä and Silvén[31], observing planar pattern as suggested by Zhang et al.[4]. The camera calibration method proposed by Zhang et al. was used to calibrate the cameras used in our experiment. The camera calibration process following [4] used a recognisable pattern attached to a planar surface, i.e., a chessboard-like pattern as used in the paper; then, the camera captured the images of the pattern from various angles. We then used the MATLAB[®] camera calibrator tool to do the computation for finding the camera parameters. At least ten images were taken following suggestions from MATLAB[®] documentation.

An 8 x 7 chessboard-like pattern with a 10 x 10 cm² square was used as a planar object to calibrate the DJI Tello drone's camera. The drone captured chessboard images from different angles, as shown in Figure 3.1. The resulting drone-mounted camera's intrinsic parameter, $\mathbf{K}_{\text{drone}_1}$, used in the ELTE experiment one, was the following:

$$\mathbf{K}_{\text{drone}_1} = \begin{bmatrix} 1718.5 & 0 & 655.2 \\ 0 & 1699.1 & 188.6 \\ 0 & 0 & 1 \end{bmatrix}$$

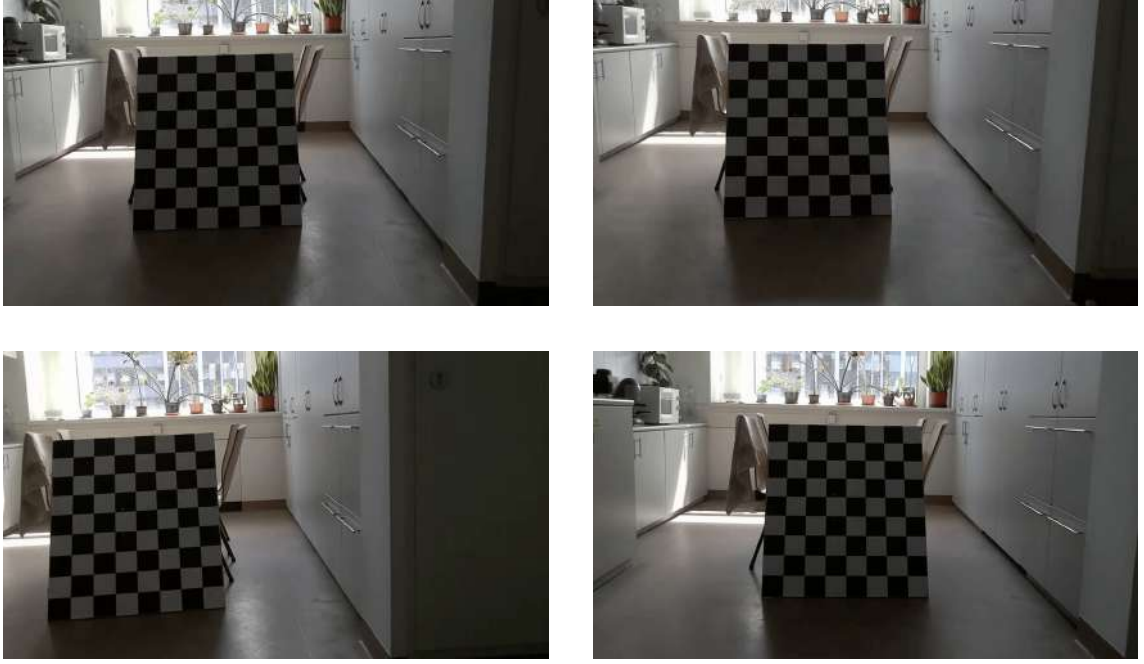


Figure 3.1: Sample of images for DJI Tello drone's camera calibration.

The DJI Phantom 4 drone's camera was calibrated using an 8 x 9 chessboard pattern, as seen in Figure 3.2. The chessboard pattern had an 8 x 8 cm² size. The camera's intrinsic parameter, $\mathbf{K}_{\text{drone}_2}$, retrieved from MATLAB[®] camera calibrator tool given as following:

$$\mathbf{K}_{\text{drone}_2} = \begin{bmatrix} 2768.8 & 0 & 1946.7 \\ 0 & 2794.9 & 993.9 \\ 0 & 0 & 1 \end{bmatrix}$$

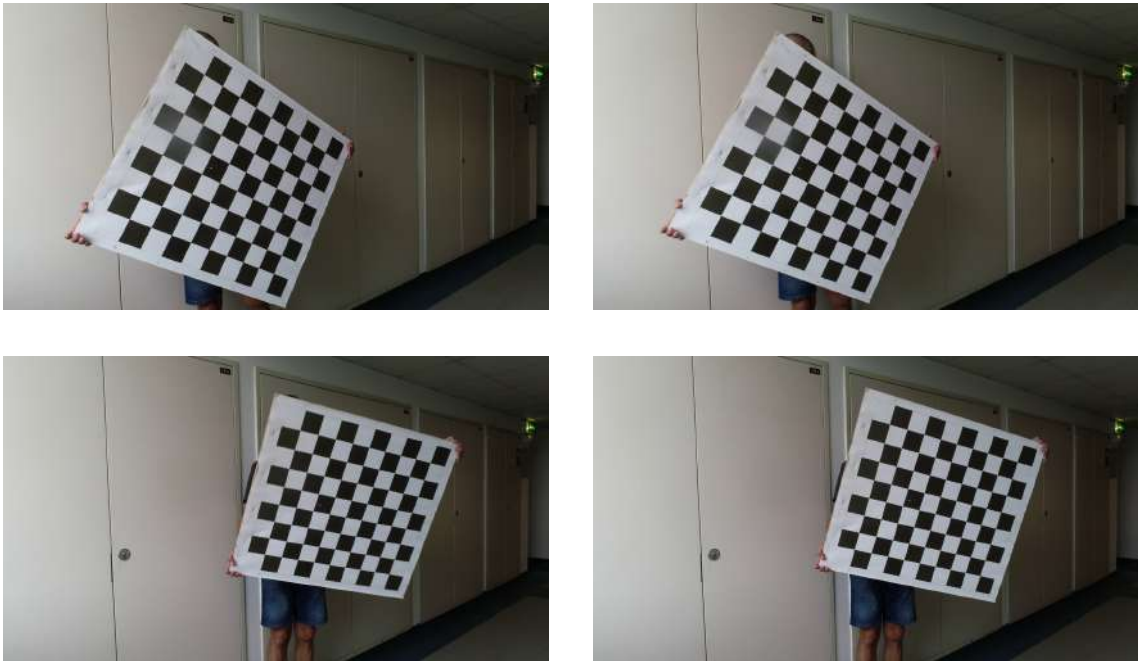


Figure 3.2: Sample of images for DJI Phantom 4 drone's camera calibration.

The car-mounted camera was already calibrated and had an intrinsic parameter, $\mathbf{K}_{\text{car}}$,

$$\mathbf{K}_{\text{car}} = \begin{bmatrix} 1326.0 & 0 & 945.5 \\ 0 & 1326.0 & 663.5 \\ 0 & 0 & 1 \end{bmatrix}$$

Meanwhile, the CARLA's camera intrinsic parameter, $\mathbf{K}_{\text{carla}}$, was derived theoretically from FOV and image resolution. Since the given FOV value in CARLA was horizontal FOV, the *image width* was used in Equation 3.3. The focal length was then used to compose the camera matrix as in 3.1, given the result of $\mathbf{K}_{\text{carla}}$.

$$f_x = f_y = \frac{\text{image_width}}{2 \cdot \tan\left(\text{FOV} \cdot \frac{\pi}{360}\right)} \quad (3.3)$$

$$\mathbf{K}_{\text{carla}} = \begin{bmatrix} 672.199 & 0 & 960 \\ 0 & 672.199 & 540 \\ 0 & 0 & 1 \end{bmatrix}$$

3.1.2 Pose Estimation

Computing the extrinsic parameter $[\mathbf{R} \mid \mathbf{t}]$ is known as the pose estimation problem[5]. The pose was determined by selecting several points in the world coordinate system and image coordinate system and then solving the transformation matrix, $\mathbf{T} = [\mathbf{R} \mid \mathbf{t}]$ that define their relations. We define points in a 3D world coordinate system and choose the same points visibly in the 2D image plane. Using n -point correspondences to determine pose was also known as Perspective- n -Point (PnP) problem[9]¹. Solving a PnP problem gave us the transformation matrix ${}^c\mathbf{T}_w$, of Equation 3.4. This equation shows a relation in the homogeneous coordinate system between a point, $P_w \begin{bmatrix} X_w & Y_w & Z_w & 1 \end{bmatrix}^T$ in the world coordinate system and a point, $P_c \begin{bmatrix} X_c & Y_c & Z_c & 1 \end{bmatrix}^T$ in the camera coordinate system.

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = {}^c\mathbf{T}_w \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \quad (3.4)$$

¹A more detailed explanation about PnP was provided in Appendix A

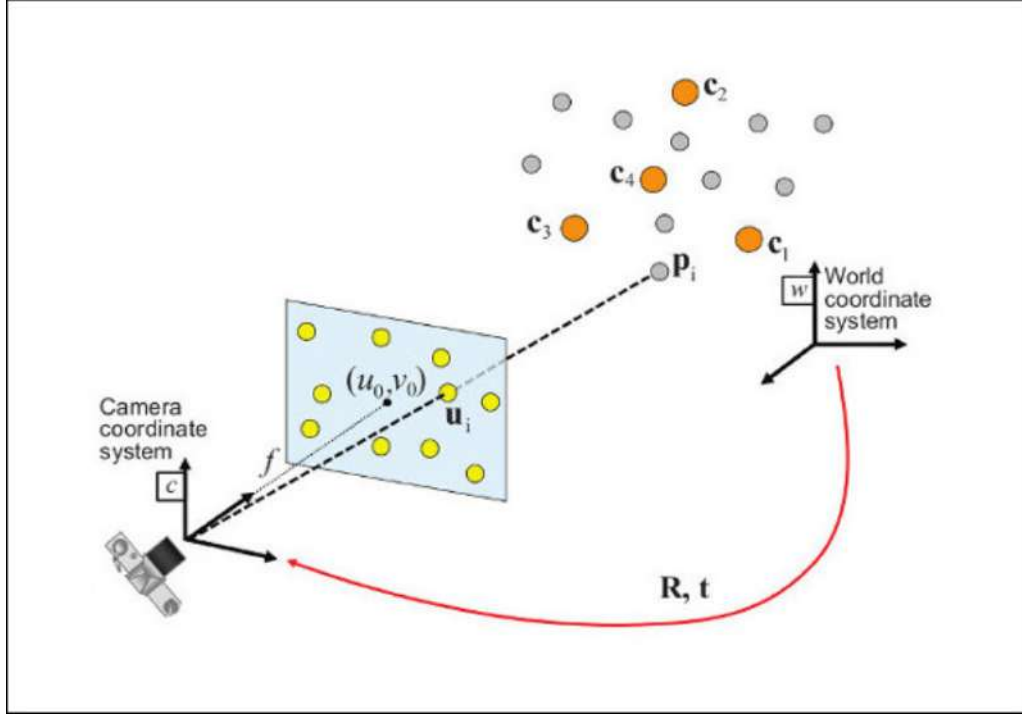


Figure 3.3: Pose estimation illustration taken from OpenCV documentation.

3.1.3 Homography

Correspondences between points in two images that reference the same object points in the world coordinate system can be defined by the homography matrix[14][26]. A point $\begin{bmatrix} x & y \end{bmatrix}^T$ in image one can be projected into point $\begin{bmatrix} x' & y' \end{bmatrix}^T$ in image two up to scale, s , by a homography matrix, $\mathbf{H}$, as shown by the Equation 3.5 and illustrated in Figure 3.4

$$s \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \mathbf{H} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (3.5)$$

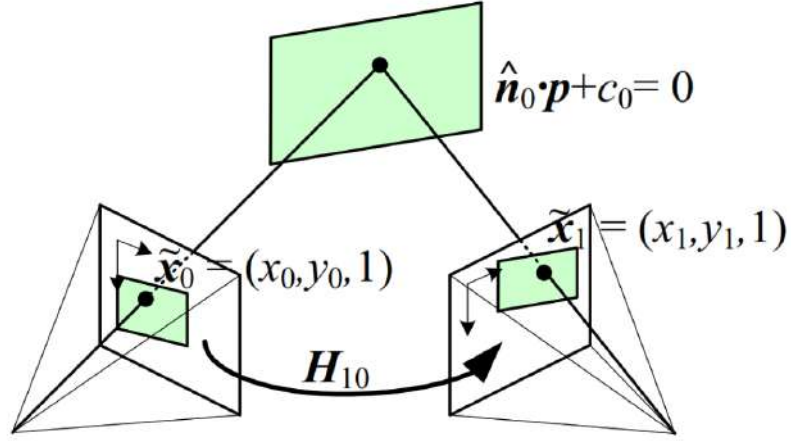


Figure 3.4: Planar homography from points lying on a common planar plane
 $\hat{n}_0 \cdot p + c_0 = 0$. [26]

From the extrinsic parameter of camera one, $[\mathbf{R}_1 \mid \mathbf{t}_1]$, the common planar plane can be expressed as its normal, n_1 , and distance, d_1 , from camera one origin to the plane. Furthermore, the homography transformation, ${}^2\mathbf{H}_1$, between image one and image two can be reconstructed as follows:²

$${}^2\mathbf{H}_1 = {}^2\mathbf{R}_1 + \frac{{}^2\mathbf{t}_1 \cdot n_1^\top}{d_1} \quad (3.6)$$

3.2 Algorithm Development

Our proposal is illustrated in Figure 3.5. In the first frame of the image, there was a common object seen by both car-mounted camera (camera 1) and drone-mounted camera (camera 2). The pose of each camera to the object reference was estimated. The resulting transformation matrices were combined to compute the pose of camera two from camera one, ${}^2\mathbf{T}_1$. Based on the pose of camera one, we define the road as a planar plane reference.

The detected chessboard on the car's roof was used as another point of correspondence to estimate the pose of the drone-mounted camera from the vehicle, which was solved as a PnP problem. The resulting transformation matrix from chessboard to drone, ${}^D\mathbf{T}_C$, combined with ${}^2\mathbf{T}_1$, determined the transformation matrix from the chessboard to the vehicle, ${}^V\mathbf{T}_C$. Since the car-mounted camera was fixed to its place, it was used as a camera reference, and the value of ${}^V\mathbf{T}_C$ was fixed even when the car moved. The position

²The following equation followed the work of Faugeras[14] presented in the Appendix B

between camera one and the road (ground) was assumed to be fixed following the same premise. Therefore, the reference plane remained the same during the whole frame. Concatenating ${}^D\mathbf{T}_C$ and ${}^V\mathbf{T}_C$ yields the result of ${}^D\mathbf{T}_V$, a transformation matrix from the camera on the vehicle to the camera on the drone. Furthermore, the homography matrix from the drone-mounted camera's image into the car-mounted camera's frame, ${}^V\mathbf{H}_D$ was the inverse of homography matrix, ${}^D\mathbf{H}_V$ derived from the transformation matrix, ${}^D\mathbf{T}_V$. As the car moved and the drone followed it, the chessboard pattern on the car's roof was detected to update the value of ${}^D\mathbf{T}_C$ and the value of ${}^D\mathbf{T}_V$, as well as the homography matrices, ${}^D\mathbf{T}_V$ and ${}^V\mathbf{D}_V$.

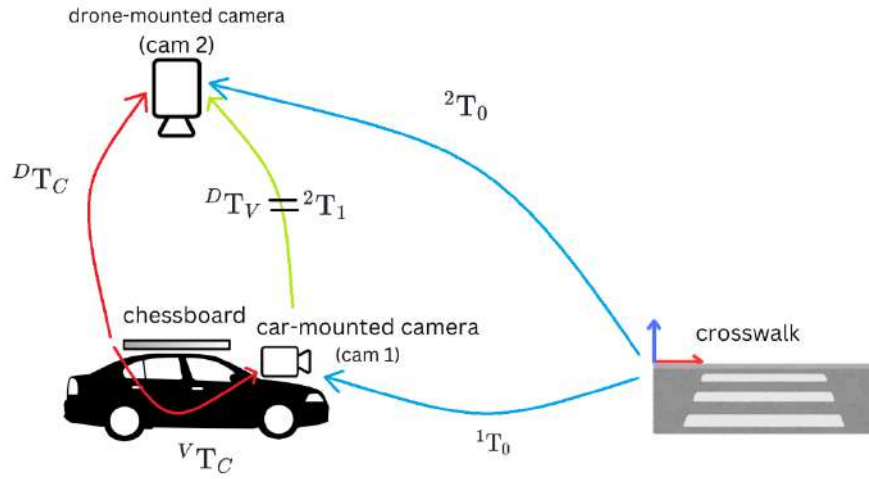


Figure 3.5: The Proposed Method

The proposed method was divided into two parts. The alignment of the car-mounted and drone-mounted cameras was explained in Algorithm 1. This step-by-step process showed how the vehicle's camera and the drone's camera were calibrated against each other. Algorithm 2 described how we updated the homography matrix between the car-mounted and drone-mounted cameras in the adjacent frames. The transformed images from each frame were combined as a video.

Algorithm 1 Alignment of Car-Mounted and Drone-Mounted Cameras

Input: Image 1 from a car-mounted camera; Image 2 from a drone-mounted camera.

- 1: Mark an object as a planar reference.
 - 2: Estimate the pose of the car-mounted camera (Camera 1). $\triangleright {}^1\mathbf{T}_0$
 - 3: Calculated road plane. $\triangleright n_1, d_1$
 - 4: Estimate the pose of the drone-mounted camera (Camera 2). $\triangleright {}^2\mathbf{T}_0$
 - 5: Concatenate ${}^1\mathbf{T}_0$ and ${}^2\mathbf{T}_0$ $\triangleright {}^2\mathbf{T}_1$
 - 6: Compute homography and its inverse using ${}^2\mathbf{T}_1$. $\triangleright {}^2\mathbf{H}_1, {}^1\mathbf{H}_2$
 - 7: Detect the chessboard on the car's roof.
 - 8: Estimate the pose of camera two from the chessboard. $\triangleright {}^D\mathbf{T}_C$
 - 9: Compute the pose of camera one from the chessboard. $\triangleright {}^V\mathbf{T}_C$
 - 10: **return** ${}^V\mathbf{T}_C = [{}^V\mathbf{R}_C \mid {}^V\mathbf{t}_C], n_1, d_1$
-

Algorithm 2 Updating Homography

Input: Image Sequence from a drone-mounted camera, ${}^V\mathbf{R}_C, {}^V\mathbf{t}_C$

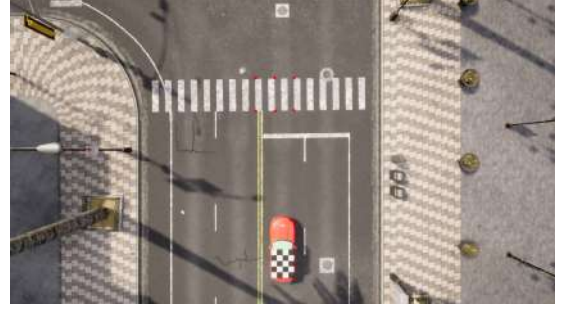
- 1: **for** each frame **do**
 - 2: Detect the chessboard on the car's roof.
 - 3: Estimate the pose of the drone-mounted camera. $\triangleright {}^D\mathbf{T}_C = [{}^D\mathbf{R}_C \mid {}^D\mathbf{t}_C]$
 - 4: Concatenate ${}^D\mathbf{T}_C$ and ${}^V\mathbf{T}_C$ $\triangleright {}^D\mathbf{T}_V$
 - 5: Compute homography matrix. $\triangleright {}^V\mathbf{H}_D$
 - 6: Perspective transformation from drone-mounted camera's frame into car-mounted camera's frame.
 - 7: **end for**
 - 8: **return** A video sequence of the transformed drone-mounted camera's image.
-

3.2.1 Pose Estimation on Each Camera

The crosswalk shown in Figure 2.14 was used as an object reference. The open-source software library, OpenCV, was used to solve the PnP problem. The OpenCV solved the PnP problem iteratively using Direct Linear Transform (DLT)[6] and Levenberg-Marquardt minimisation[5]. Initial solutions need six correspondence points between the image and the world object. Therefore, as seen in Figure 3.6, six points were selected manually on the car-mounted and drone-mounted cameras' images as 2D image points. The 3D object points in the world coordinate system were defined from the crosswalk dimension in Figure 2.14. These point correspondences were used to solve the 3D-2D projection using a PnP algorithm, which gave a result of rotation vector, r_{vecs} and translation vector, t_{vecs} .



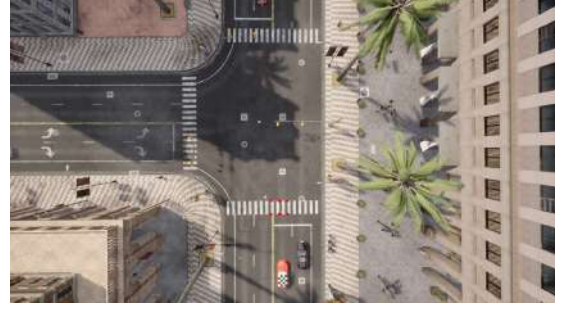
(a) Frontal View Image



(b) 15 meters bird-eye-view



(c) 25 meters bird-eye-view



(d) 35 meters bird-eye-view

Figure 3.6: Six points (red dots) were selected on the crosswalk as planar reference.

Subsequently, the road plane was determined to be a planar reference. The normal of the plane, n_1 , was computed from the rotation matrix of camera one, $\mathbf{R}_1$, times the unit normal in the z-direction. We determined the z-direction as a vector pointing upward, as can be seen in Figure 3.7, respectively, using the right-hand coordinate system, the x-axis was in accordance with the car's forward direction (blue arrow), and the y-axis pointed to the left direction (green arrow). Since we knew the translation from the origin of the world coordinate system to the origin of the camera coordinate system, t_{vecs} , the plane distance, d_1 , could be easily retrieved from it. Camera one was fixed to the vehicle; therefore, it had a fixed position relative to the road. In this regard, the planar plane computed in the first frame was used continuously for the subsequent frame.



Figure 3.7: The coordinate system on the road plane.

The origin of the world coordinate system can be drawn in the image by OpenCV, as shown in Figure 3.8 and Figure 3.9, by using the value of r_{vecs} and t_{vecs} .



(a) Pose from the car-mounted camera.



(b) 15 meters drone-mounted camera.



(c) 25 meters drone-mounted camera.



(d) 35 meters drone-mounted camera.

Figure 3.8: Origin of the world coordinate system in image frame.



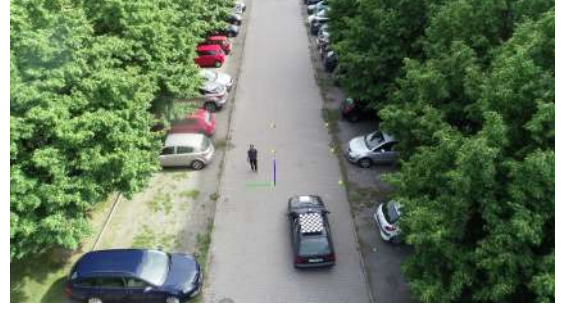
(a) Pose of the car-mounted camera of experiment one.



(b) Pose of drone-mounted camera of experiment one.



(c) Pose of the car-mounted camera of experiment two.



(d) Pose of the drone-mounted camera of experiment two.

Figure 3.9: Pose estimation in the real world experiment.

3.2.2 Concatenating Pose Between Cameras

Solving the PnP problem yields the transformation matrix, $\mathbf{T}$, a key component in our methodology. Specifically, ${}^1\mathbf{T}_0$ represents the transformation matrix from the world coordinate system to the car-mounted camera (camera one's coordinate system), and ${}^2\mathbf{T}_0$ represents the transformation matrix from the world coordinate system to the drone-mounted camera (camera two's coordinate system).

$${}^1\mathbf{T}_0 = \begin{bmatrix} {}^1\mathbf{R}_0 & {}^1\mathbf{t}_0 \\ \mathbf{0}_{3 \times 1} & 1 \end{bmatrix} \quad (3.7)$$

$${}^2\mathbf{T}_0 = \begin{bmatrix} {}^2\mathbf{R}_0 & {}^2\mathbf{t}_0 \\ \mathbf{0}_{3 \times 1} & 1 \end{bmatrix} \quad (3.8)$$

Combining 3.7 and 3.8, the transformation matrix from camera one's coordinate into camera two's coordinate, ${}^2\mathbf{T}_1$, was calculated using Equation 3.9. Given the plane normal, n_1 and the distance, d_1 from camera one into the planar reference, the homography matrix from camera one into camera two, ${}^2\mathbf{H}_1$ was calculated using Equation 3.6[14].

$${}^2\mathbf{T}_1 = {}^2\mathbf{T}_0 {}^0\mathbf{T}_1 = {}^2\mathbf{T}_0 ({}^1\mathbf{T}_0)^{-1} = \begin{bmatrix} {}^2\mathbf{R}_0 & {}^2\mathbf{t}_0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} {}^1\mathbf{R}_0^T & -{}^1\mathbf{R}_0^T {}^1\mathbf{t}_0 \\ 0 & 1 \end{bmatrix} \quad (3.9)$$

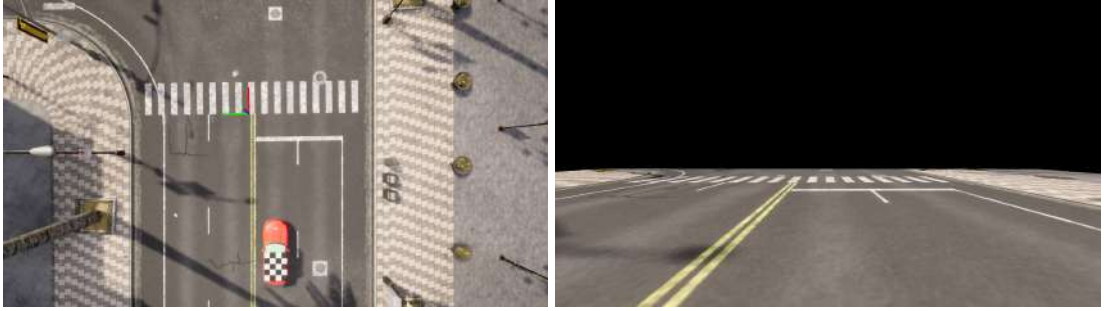
Later on, the homography from camera two's frame to camera one's frame, ${}^1\mathbf{H}_2$ was the inverse of the homography matrix, ${}^2\mathbf{H}_1$.

$${}^1\mathbf{H}_2 = ({}^2\mathbf{H}_1)^{-1} \quad (3.10)$$

Given the camera matrix, $\mathbf{K}$, the projective homography matrix, $\mathbf{G}$, can be calculated as following equation:

$$\mathbf{G} = \mathbf{K}\mathbf{H}\mathbf{K}^{-1} \quad (3.11)$$

The projective homography matrix, ${}^1\mathbf{G}_2$, was then applied to transform camera two's image into the camera one's coordinates system, resulting in a nice perspective transformation as depicted in Figure 3.10 for the CARLA's generated image and Figure 3.12 for the ELTE's experiment.



(a) Drone at 15 meters of height



(b) Drone at 25 meters of height



(c) Drone at 35 meters of height

Figure 3.10: Drone-mounted camera image (left) and its projective transformation result (right)



(a) Drone at 15 meters of height



(b) Drone at 25 meters of height

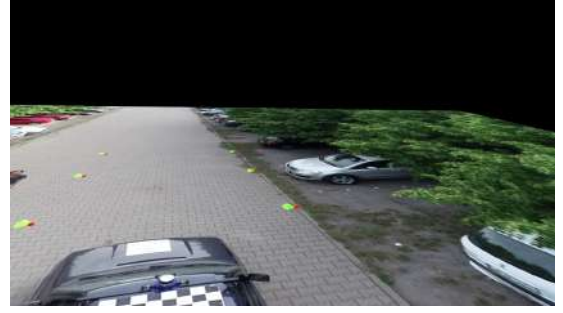


(c) Drone at 35 meters of height

Figure 3.11: Comparison of the projective transformation result (left) with the image taken by the car-mounted camera (right)



(a) Image from the drone-mounted camera.



(b) Projective transformation from the drone-mounted camera's image.



(c) Image from the car-mounted camera.

Figure 3.12: Left: experiment one, right: experiment two

3.2.3 Chessboard Detection

The chessboard pattern on the car's roof was detected using OpenCV. By utilising the known dimension of the chessboard in the world coordinate system, the chessboard's corners were used as 3D-2D point pairs to solve the PnP problem to estimate the pose of the drone-mounted camera (d : *drone*) from the chessboard (c : *chessboard*), a crucial step in our research. The resulting transformation matrix, ${}^D\mathbf{T}_C$, was used to draw the origin of the world coordinate system in the image. The chessboard detection worked in the various altitudes we tried in CARLA, as seen in Figure 3.13 to Figure 3.15.



Figure 3.13: Drone pose at an altitude of 15 meters.

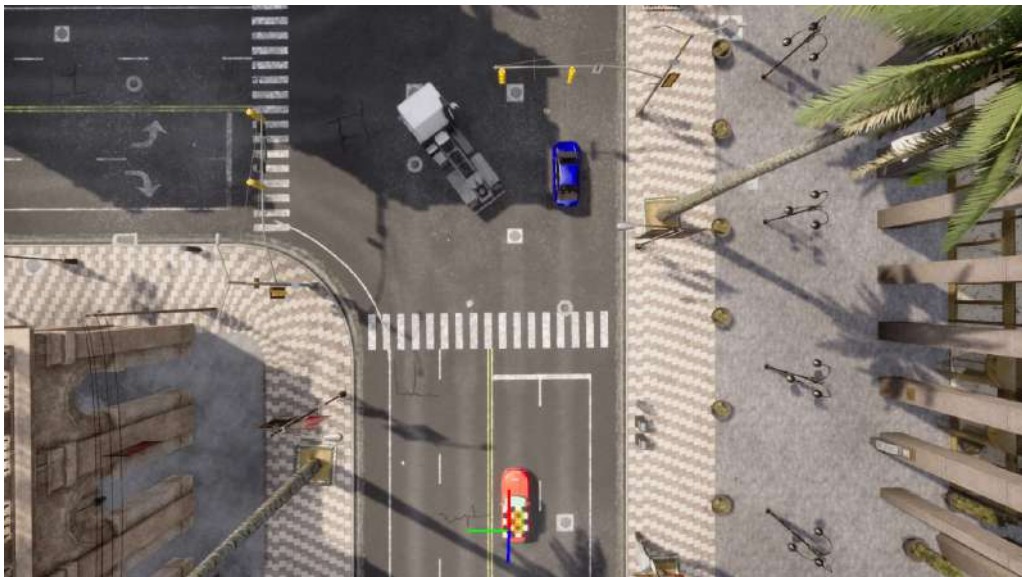


Figure 3.14: Drone pose at an altitude of 25 meters.



Figure 3.15: Drone pose at an altitude of 35 meters.

In the real-world experiment in Figure 3.16 and Figure 3.17, we observed that OpenCV's chessboard detection could work to identify the chessboard from various angles. The chessboard detection even functioned properly in nearly every frame of experiment two since the resulting video was given in the 4K resolution. The provided chessboard detection from OpenCV's library could be used in the proposed algorithm.

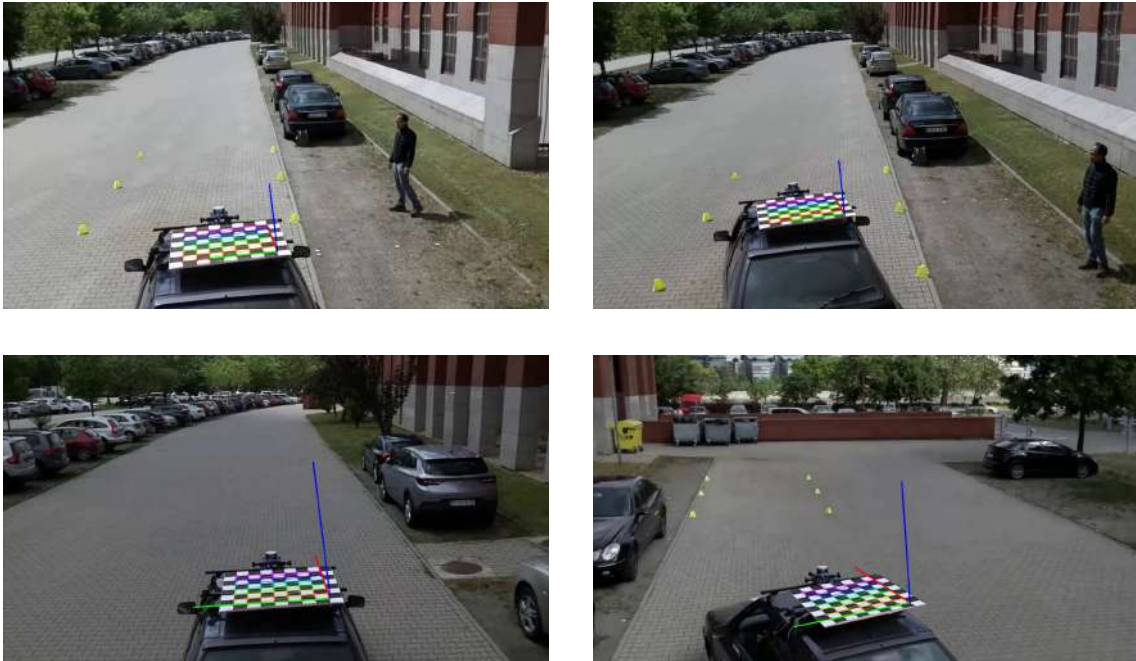


Figure 3.16: Chessboard Detection in the ELTE's Experiment 1.

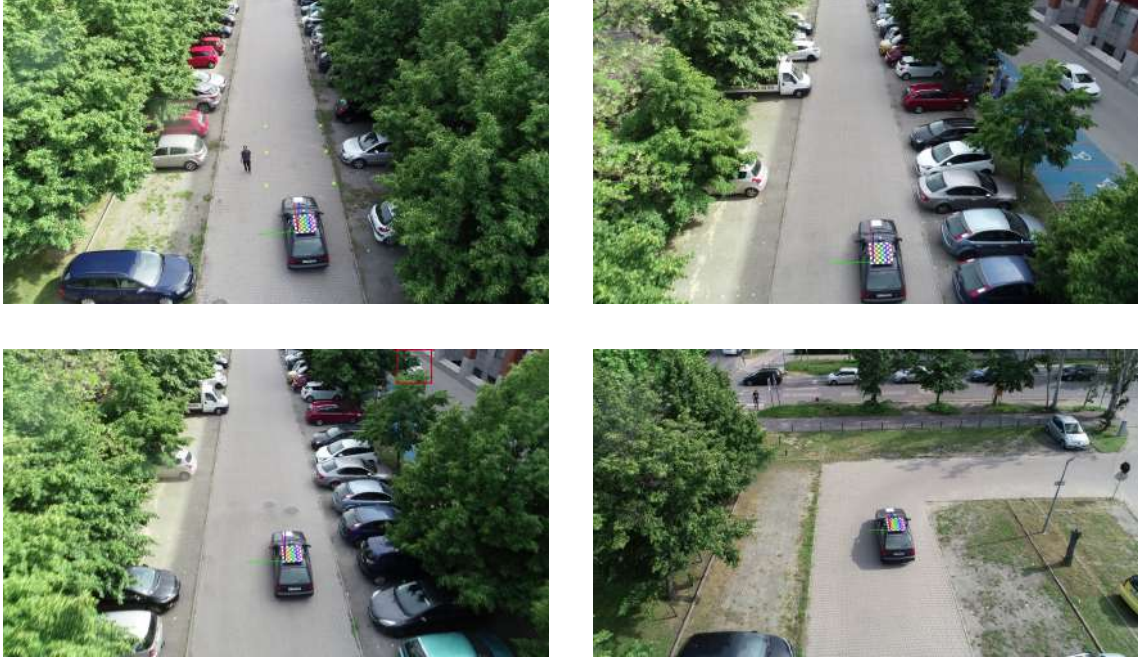


Figure 3.17: Chessboard Detection in the ELTE's Experiment 2.

3.2.4 Vehicle's Camera Pose from The Chessboard

Since the chessboard on the car's roof was out of sight of the car-mounted camera, the pose of the chessboard to the car-mounted camera (v : *vehicle*) cannot be calculated using a solve PnP algorithm. However, the pose could be estimated using a previously computed transformation matrix between cameras one and two, ${}^2\mathbf{T}_1$. The transformation matrix from the chessboard into the vehicle (car-mounted camera), ${}^v\mathbf{T}_C$ was calculated as follows:

$${}^v\mathbf{T}_C = {}^1\mathbf{T}_2 {}^D\mathbf{T}_C = ({}^2\mathbf{T}_1)^{-1} {}^D\mathbf{T}_C = \begin{bmatrix} {}^2\mathbf{R}_1^\top & -{}^2\mathbf{R}_1^\top {}^2\mathbf{t}_1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} {}^D\mathbf{R}_C & {}^D\mathbf{t}_C \\ 0 & 1 \end{bmatrix} \quad (3.12)$$

3.2.5 Relative Pose Estimation of Car-Mounted Camera from Drone-Mounted Camera

Using a similar equation as Equation 3.9, the transformation of the image from the vehicle's camera into the drone's camera, ${}^D\mathbf{T}_V$ was computed as follows.

$${}^D\mathbf{T}_V = {}^D\mathbf{T}_C {}^C\mathbf{T}_V = {}^D\mathbf{T}_C ({}^v\mathbf{T}_C)^{-1} = \begin{bmatrix} {}^D\mathbf{R}_C & {}^D\mathbf{t}_C \\ 0 & 1 \end{bmatrix} \begin{bmatrix} {}^v\mathbf{R}_C^\top & -{}^v\mathbf{R}_C^\top {}^v\mathbf{t}_C \\ 0 & 1 \end{bmatrix} \quad (3.13)$$

We have proven that the transformation matrix ${}^2\mathbf{T}_1$ from Equation 3.9 was the same as ${}^D\mathbf{T}_V$ from Equation 3.13. It can also be seen in Figure 3.5 that this was the closed-loop vector relation. Using the plane normal, n_1 , and distance, d_1 , the homography matrix ${}^D\mathbf{H}_V$ can be derived from ${}^2\mathbf{T}_1$ using Equation 3.6. The projective homography matrix, ${}^V\mathbf{G}_D$, was used to project the drone-mounted camera's image into the car-mounted camera's frame, like in Figure 3.18 as an example. Because the transformation matrices were the same and using the road as a planar reference, the image projection was the same either using the projective homography matrix, ${}^1\mathbf{G}_2$ or ${}^V\mathbf{G}_D$.

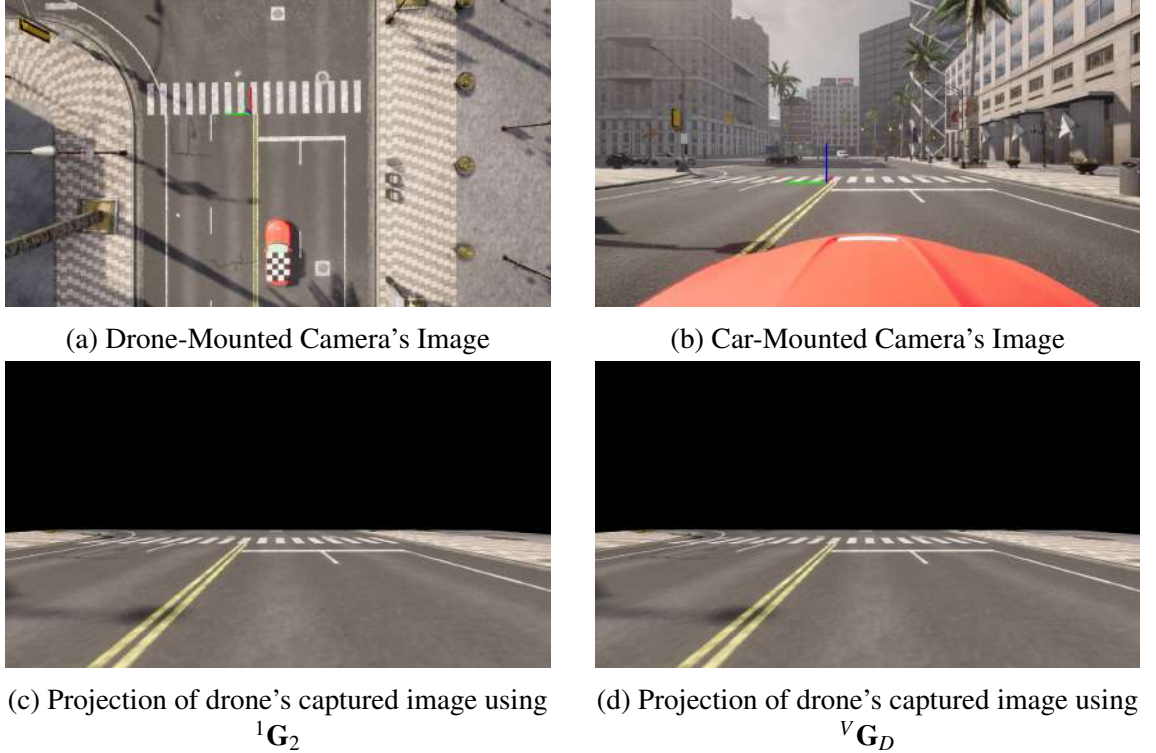


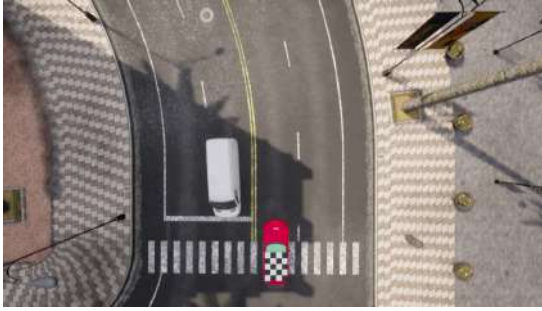
Figure 3.18: Comparison of the homography matrix derived from ${}^1\mathbf{T}_2$ and ${}^V\mathbf{T}_D$.

3.2.6 Homography Matrix Composition

As the car and the drone moved, the crosswalk on the road would not always be visible in both cameras. Therefore, the process of tracking the chessboard was used. The chessboard was detected using the OpenCV. The detected chessboard's points were used to solve the PnP problem and estimate the pose of the drone-mounted camera, ${}^D\mathbf{T}_C$, in that frame. The transformation matrix, ${}^D\mathbf{T}_C$, was updated, while the transformation matrix, ${}^V\mathbf{T}_C$, remained the same in each frame. The process would then update the value of ${}^D\mathbf{T}_V$, using the Equation 3.13. The transformation matrix, ${}^D\mathbf{T}_V$, consists of rotation matrix, ${}^D\mathbf{R}_V$, and translation vector, ${}^D\mathbf{t}_V$.

$${}^D\mathbf{T}_V = \begin{bmatrix} {}^D\mathbf{R}_V & {}^D\mathbf{t}_V \\ \mathbf{0}_{3 \times 1} & 1 \end{bmatrix} \quad (3.14)$$

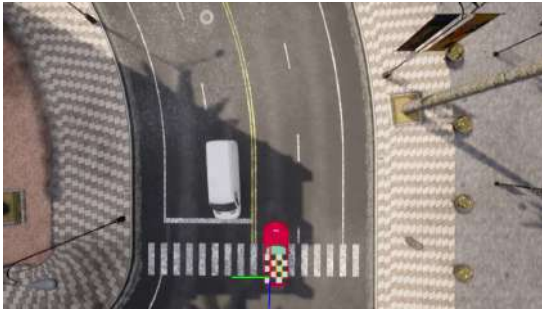
Given the value of ${}^D\mathbf{R}_V$ and ${}^D\mathbf{t}_V$, the homography matrix ${}^D\mathbf{H}_V$ between vehicle and drone was calculated using the Equation 3.6. Following Equation 3.10 and Equation 3.11, the projective homography matrix from the drone-mounted camera into the car-mounted camera, ${}^V\mathbf{G}_D$, was composed. Figure 3.19 to 3.21 showed the projection of the drone-mounted camera's image from the CARLA in the adjacent frame.



(a) Drone-mounted camera's image.



(b) Car-mounted camera's image.

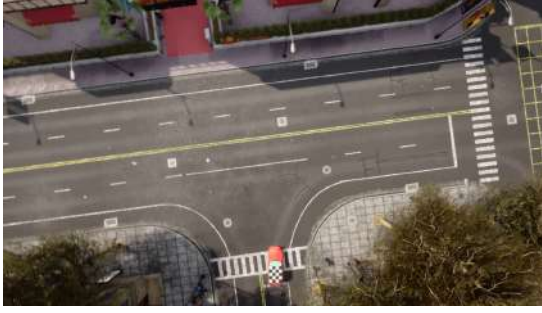


(c) Chessboard detection.



(d) Perspective transformation.

Figure 3.19: Perspective transform of 15-meter-high drone camera's image.



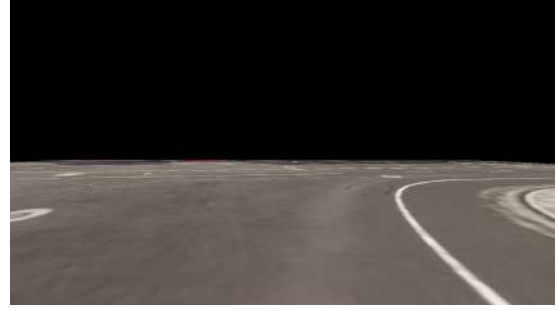
(a) Drone-mounted camera's image.



(b) Car-mounted camera's image.



(c) Chessboard detection.



(d) Perspective transformation.

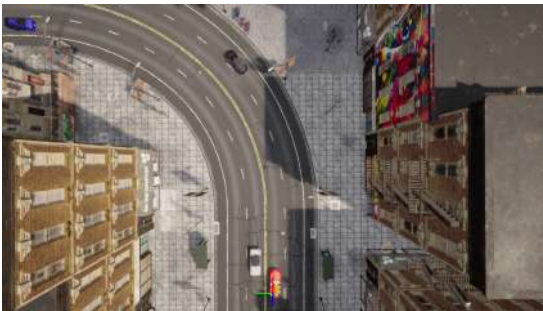
Figure 3.20: Perspective transform of 25-meter-high drone camera's image.



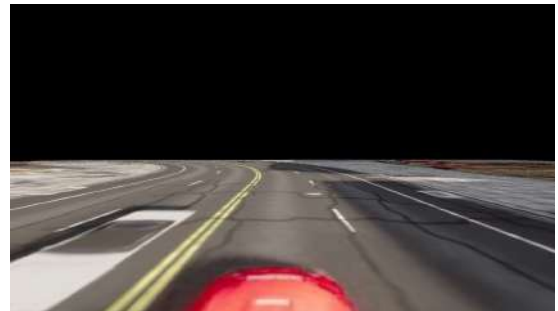
(a) Drone-mounted camera's image.



(b) Car-mounted camera's image.



(c) Chessboard detection.



(d) Perspective transformation.

Figure 3.21: Perspective transform of 35-meter-high drone camera's image.

Different types of cameras were used in the ELTE experiments. Therefore, Equation 3.11 was adjusted accordingly to the camera matrices $\mathbf{K}_{car}$, $\mathbf{K}_{drone1}$, and $\mathbf{K}_{drone2}$. The selection of the camera matrix depended on which image we would like to transform.

The transformation from the ELTE's experiment was presented in Figure 3.22 and Figure 3.23



(a) Drone-mounted camera's image.



(b) Car-mounted camera's image.



(c) Chessboard detection.



(d) Perspective transformation.

Figure 3.22: Perspective transform of ELTE Experiment 1.



(a) Drone-mounted camera's image.



(b) Car-mounted camera's image.



(c) Chessboard detection.



(d) Perspective transformation.

Figure 3.23: Perspective transform of ELTE Experiment 2.

3.3 Result and Discussion

The proposed algorithm has been tested in the 12 scenarios of CARLA-generated videos and confirmed to function effectively. First, the drone flew perfectly above the car and followed exactly the movement of the car. We tested this setup at three different altitudes: 15 meters, 25 meters, and 35 meters. The results shown in Figure 3.19 to Figure 3.21 have confirmed that the algorithm could successfully transform the image with excellent visible results even when the drone is flying as high as 35 meters above the ground. These outcomes were particularly noteworthy as they demonstrated the algorithm's ability to function optimally despite being subjected to different altitude levels.

Afterwards, we simulated some possible disturbances on the drone at each altitude. The drone had a roll movement over time to simulate the effect of rotation. Then, we modified the longitudinal position of the drone to simulate the effect of translation. And finally both rotation and translation movement was simulated in the drone-mounted camera. In each scenario, the transformed image was combined sequentially as a video.

Notably, the image projection produced high-quality results, even in the presence of disturbances like rotation in the drone, as depicted in Figure 3.24 to Figure 3.26. The proposed algorithm could also successfully accommodate the relative translation movement between drone and vehicle. Presenting the translation movement in a static image was difficult. Therefore, we have compiled all the videos result in the designated [*folder*](#)³ for further analysis and review. In conclusion, the proposed algorithm has demonstrated its ability to function correctly across different altitudes and distortions. Following this premise, we carried away the test into the experiment setup in the real world at the ELTE's parking space. Two real-world data experiments were conducted, and the proposed algorithm was successfully implemented on both videos captured using different types of drones.

³The video result of this work can be accessed in the following [OneDrive](#)

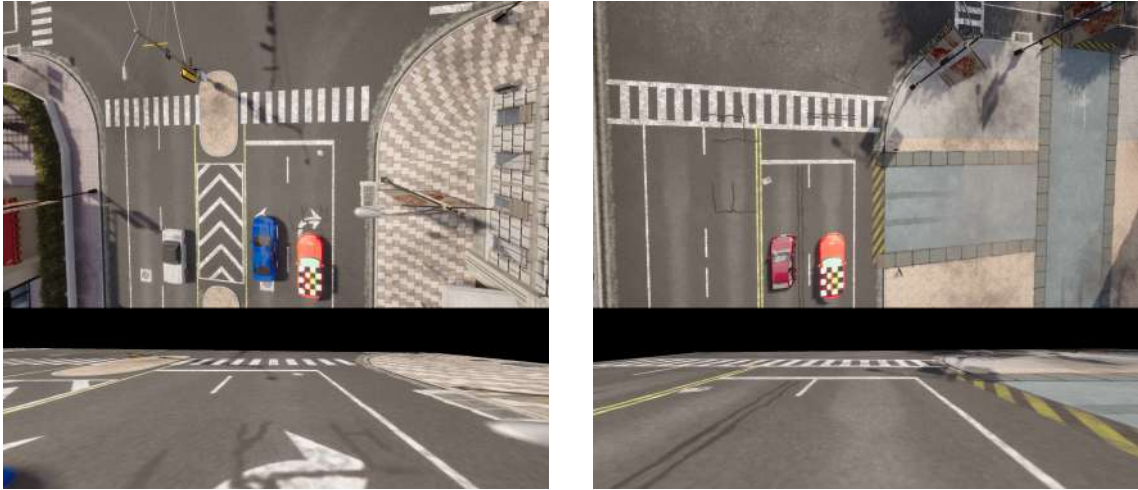


Figure 3.24: 15-meter high drone image with rotation.

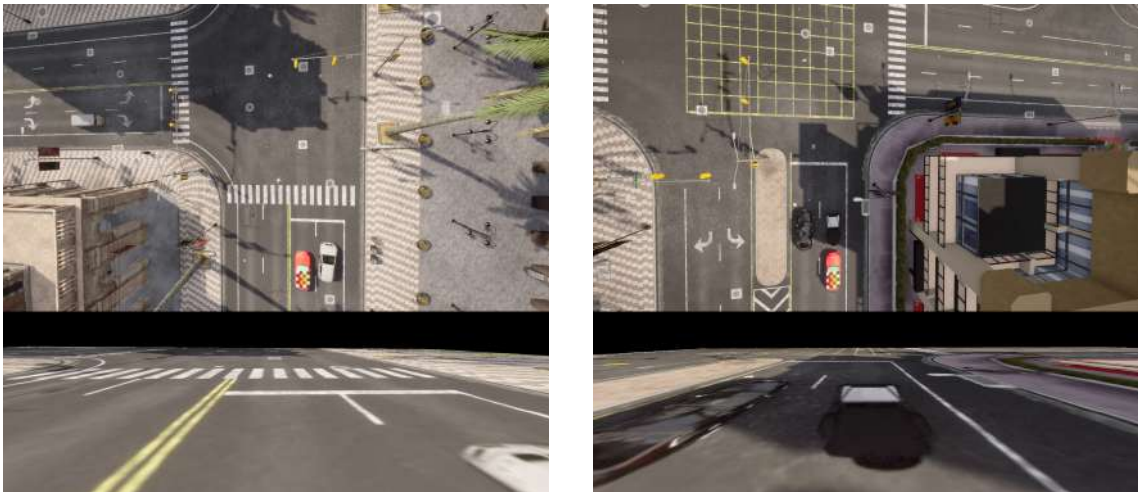


Figure 3.25: 25-meter high drone image with rotation.



Figure 3.26: 35-meter high drone image with rotation.

We tested the proposed algorithm in the real-world dataset. The data acquisition process was conducted in ELTE's parking place. In the resulting video³, we saw that without a stabiliser, the effect of wind on the drone was prominent, like in the first experiment with DJI Tello. The shaking effect in the video captured by the DJI Tello drone was recognisable. The proposed algorithm could still give a good-quality image transformation even under such disturbance.

We found that the variations in the cameras mounted on a car and a drone, such as different FOVs, as we tested in the second experiment, enriched us with more information. This helped us gain a different point of view on the same traffic situation. The image captured by the drone-mounted camera offered a perspective that was not easily recognisable by the car-mounted camera. In Figure 3.27, an image captured by the car-mounted camera, it was difficult to identify a car approaching from the other side. However, in the image captured by the drone-mounted camera, as shown in Figure 3.28, the vehicle, as indicated by the red box, could be identified. This highlights the importance of having multiple perspectives on traffic situations, especially when object identification is needed.

By transforming the drone-mounted camera's image into the car-mounted camera's coordinate system, we could approximately locate the car's position in the camera's frame, as seen in Figure 3.29, a green arrow has been added to emphasise the result. Access to various perspectives through a drone-mounted or other camera at different locations can help better understand and analyse a traffic situation.



Figure 3.27: Image taken by the car-mounted camera.



Figure 3.28: Red box showed the location of an identified car.



Figure 3.29: Perspective transformation showed the car's location (green arrow).

To further analyse our findings, we conducted tests on images generated by CARLA. We manually annotated these images, as depicted in Figure 3.30. We then used the projective homography matrix to transform these BEV images, obtaining the box location perceived in the vehicle's camera frame, as shown in Figure 3.31. We extracted the location of the transformed bounding box and used it to draw a bounding box in the ground truth image from the car-mounted camera, as presented in Figure 3.32. By identifying objects from the drone-mounted camera's point of view, the proposed algorithm can be used to locate the objects in the car-mounted camera's image.

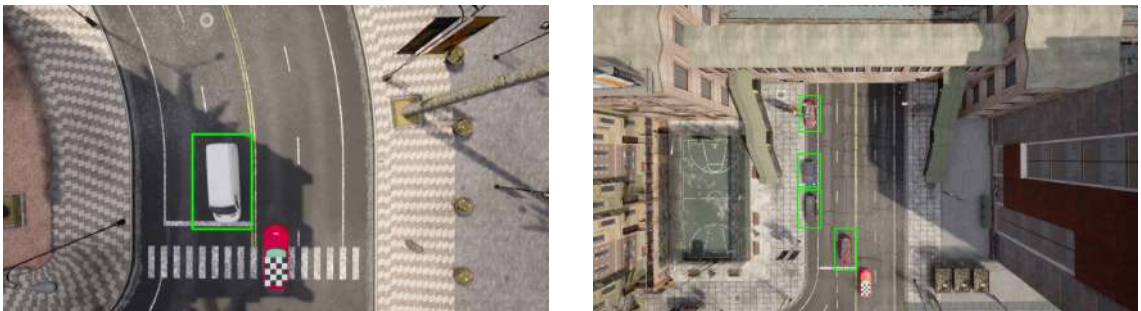


Figure 3.30: Annotated images.

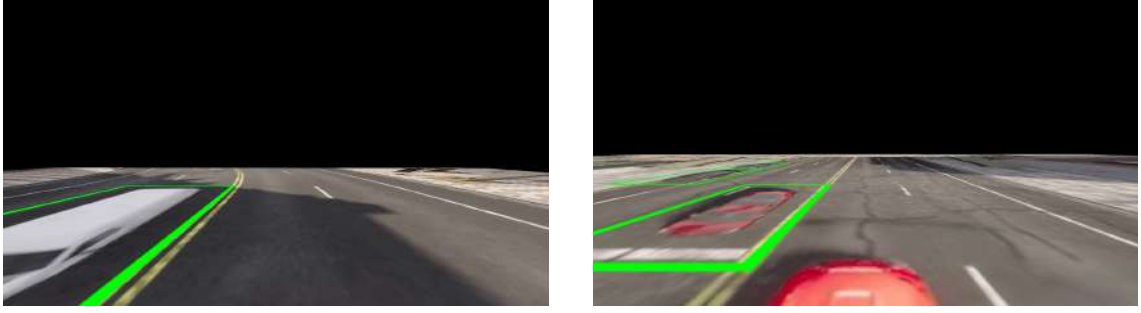


Figure 3.31: Transformed annotated images.

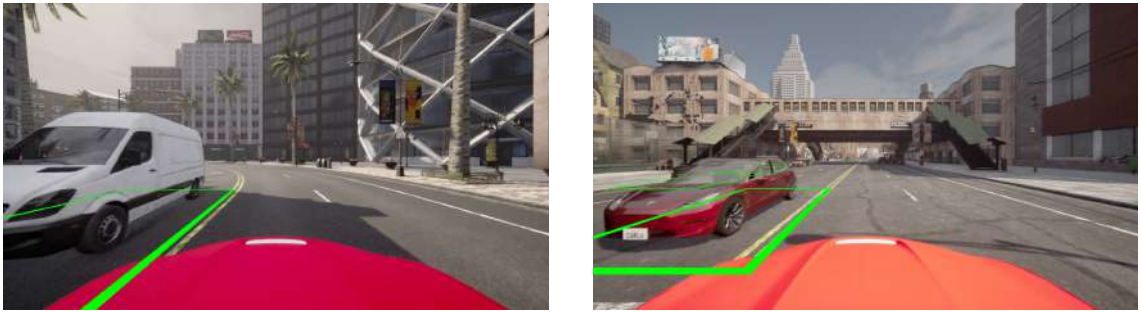


Figure 3.32: Drawing bounding box using location from transformed images.

Chapter 4

Conclusion and Future Works

An algorithm that can estimate the pose of the car-mounted camera from the respective drone-mounted camera has been developed. Using the resulting pose, the homography was computed to transform the drone-mounted camera's image into the car-mounted camera's coordinate. Several scenarios were generated in the autonomous driving car simulator, CARLA, and the algorithm's functionality has been proven.

The proposed algorithm has been tested using real-world data captured from the ELTE Car and various types of drones. The results, as detailed in Section 3.3, highlight the algorithm's ability to leverage the view from a drone-mounted camera, providing additional information from the perspective of the car-mounted camera. Notably, the algorithm demonstrated its functionality even in the presence of disturbances in drone movement. The disturbance from the wind clearly affected the video from the DJI Tello drone. Nevertheless, the proposed algorithm could manage to transform the image correctly into the car-mounted camera's coordinate.

The accuracy of the chessboard detection notably influenced the algorithm's robustness. In a very high definition (3840 x 2160), like the one generated by DJI Phantom, the chessboard was easily detected, even from different angles and altitudes. On the other hand, the chessboard on images generated by DJI Tello with a resolution of 1280 x 720 was a bit challenging to detect. Implementing a more robust chessboard detection is a potential area for future research which could significantly enhance the algorithm's performance. Benefiting from developing a motion tracking algorithm to determine pose estimation, we could also broaden the vehicle tracking method to include more than just the chessboard pattern object.

Object detection algorithms, such as YOLO[32], could also be added to the algorithm.

As discussed earlier, some objects were more visible in BEV than in the usual viewpoint of the car-mounted camera. Automatically detecting the objects and transforming them into a car camera's coordinate system will help to gain more information about the surrounding area.

A time-synchronised video between a drone and a car was generated using the CARLA simulator. This was a perfect condition for the experiment since we could quickly identify the image pairs between a car-mounted and a drone-mounted camera, which was not the case in a real-world experiment. This was reserved for future work to synchronise the time when the image was captured using a car-mounted and, at the roughly exact moment, a drone-mounted camera. One possibility could be a Bluetooth or WI-FI connection between a drone and a car that sends a signal to each other when the recording starts in one camera and triggers the recording in another camera.

Moreover, it would be more useful if the transformation could be done online instead of waiting until the data acquisition process is finished and carried away as a post-processing process. The thesis work was limited to only proof of the algorithm's feasibility without addressing its computational performance. The fast computational programming and integration of the algorithm as an embedded system that runs locally in a drone's or car's processor can be one area of future development.

Summarising the thesis, the algorithm to estimate the pose of a car-mounted camera in a traffic situation from a drone-mounted camera's image has been developed and proven to work as expected. This opened up some other future research mentioned earlier for the improvement and exploration of the usage of aerial images in developing the autonomous vehicle system.

Acknowledgements

I would like to give my gratitude to the Ministry of Communication and Informatics, Republic of Indonesia, who funded my study to help me pursue a higher education. I specially attributed this work to Robert Bosch, Kft., who funded the thesis work and made this research possible to happen.

I would like to offer my very special thanks to Dr. Hajder Levente, my supervisor, who introduced me to interesting computer vision research and for his kindness, patient guidance, enthusiastic encouragement, help, and mentoring in this thesis work. My appreciation also went to Bandó Kovács, Tarlan Ahadli, and Dr Márton Pál, who helped me a lot by lending their device and capturing the data for the real-world experiment. My thanks are extended to Emmanuella Helen, who assisted me a lot in learning Blender and creating the car model for simulation.

Appendix A

Perspective- n -Points

The goal of pose estimation is to recover the object's 3D pose from a set of 2D point projections. It is also known as an extrinsic calibration of the camera parameters[26]. Given a set of N points $\mathbf{X}_i, i \in \{1, \dots, N\}$ in the world coordinate system, its projection to the point set $\mathbf{x}_i, i \in \{1, \dots, N\}$ in the image plane can be known from the relation A.1[5].

$$\begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \mathbf{K}\Pi[\mathbf{R} \mid \mathbf{t}] \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix} \quad (\text{A.1})$$

where $\mathbf{K}$ is the camera intrinsic parameter

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

and Π is the projection matrix; in the case of the perspective projection model, Π is given by

$$\Pi = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Define the transformation matrix $\mathbf{T} = [\mathbf{R} \mid \mathbf{t}]$. Given 2D-3D point correspondences, $\mathbf{x}_i$ and $\mathbf{X}_i$, pose estimation aims to solve Equation A.1, and then recover the rigid transformation, $\mathbf{T}$. The minimum point correspondences needed to solve this relation are three, which is known as *perspective-3-point-problem* (P3P) as suggested by [8][33][34], and extended to the large number of points n , known as *perspective- n -point-problem* (P n P)[8][9][33] [35].

A.1 Linear Algorithms

One of the common methods to recover pose is to solve the linear equation of point projection directly; therefore, it was called *direct linear transform* with the in-depth explanation can be seen in [6][7][26]. The transformation matrix from the world coordinate system into the camera coordinate system, ${}^c\mathbf{T}_w$, can be defined as a homogeneous matrix[5].

$${}^c\mathbf{T}_w = \begin{pmatrix} \mathbf{r}_1 & t_x \\ \mathbf{r}_2 & t_y \\ \mathbf{r}_3 & t_z \\ \mathbf{0}_{3 \times 1} & 1 \end{pmatrix}$$

Following [5] and [6], given several point correspondences, we can develop Equation A.1 into the system of linear equations.

$$\mathbf{A}\mathbf{h} = \begin{pmatrix} \vdots \\ \mathbf{A}_i \\ \vdots \end{pmatrix} \mathbf{h} = 0$$

where,

$$\mathbf{A} = \begin{bmatrix} {}^wX_i & {}^wY_i & {}^wZ_i & 1 & 0 & 0 & 0 & 0 & x_i {}^wX_i & -x_i {}^wY_i & -x_i {}^wZ_i & -x_i \\ 0 & 0 & 0 & 0 & {}^wX_i & {}^wY_i & {}^wZ_i & 1 & x_i {}^wX_i & -y_i {}^wY_i & -y_i {}^wZ_i & -y_i \end{bmatrix}$$

and

$$\mathbf{h} = (\mathbf{r}_1, t_x, \mathbf{r}_2, t_y, \mathbf{r}_3, t_z)^T$$

Through the Singular Value Decomposition (SVD), the solution was given by the eigenvector of $\mathbf{A}$ corresponding to its minimal eigenvalue. An orthonormalisation was then used to reconstruct the rotation matrix[5].

A.2 Iterative Non-Linear Algorithms

The linear algorithm described previously was prone to noise[5], especially because it was an over-parameterized problem. Therefore, another solution that considers the problem's non-linearity was preferred. One approach was to solve the PnP problem as a minimisation problem. An optimisation technique like Gauss-Newton of

Levenberg-Marquadt was used to minimise the norm of reprojection error. Define $\mathbf{q} \in se(3)$ as a minimal representation of ${}^c\mathbf{T}_w(\mathbf{q} = ({}^c\mathbf{t}_w, \boldsymbol{\theta}\mathbf{u})^T$ where $\boldsymbol{\theta}$ and $\mathbf{u}$ are the angle of the rotation matrix ${}^c\mathbf{R}_w$. Equation A.2 shows the problem formula,

$$\hat{\mathbf{q}} = \underset{\mathbf{q}}{\operatorname{argmin}} \sum_{i=1}^N d(\mathbf{x}_i, \Pi {}^c\mathbf{T}_w {}^w\mathbf{X}_i)^2 \quad (\text{A.2})$$

which was solved by minimising the cost function $E(\mathbf{q}) = \|\mathbf{e}(\mathbf{q})\|$ defined by

$$E(\mathbf{q}) = \mathbf{e}(\mathbf{q})^\top \mathbf{e}(\mathbf{q}), \text{ with } \mathbf{e}(\mathbf{q}) = \mathbf{x}(\mathbf{q}) - \mathbf{x} \quad (\text{A.3})$$

where $\mathbf{x}(\mathbf{q}) = (\dots, \pi({}^c\mathbf{T}_w {}^w\mathbf{X}_i), \dots)^\top$ and $\mathbf{x} = (\dots, \tilde{\mathbf{x}}_i, \dots)^\top$ where $\tilde{\mathbf{x}}_i = (x_i, y_i)$ is an Euclidian 2D point and $\pi(\mathbf{X})$ is the projection function of a 3D point $\mathbf{X}$ into $\tilde{\mathbf{x}}$

The minimisation problem was then solved using iterative least squares (ILS). A complete derivation of the presented non-linear algorithm can be read in the paper by Marchand et al.[5].

A.3 Non-Iterative Non-Linear Algorithm

An iterative approach usually needs a good initialisation. A poor initial estimation can lead to a failure in an iterative algorithm. Lepetit et al.[9] try to tackle this drawback by suggesting an algorithm called EPnP, a non-iterative solution to the PnP problem. Some other examples of a non-iterative approaches were OPnP suggested by Zheng et al.[36] and GPnP proposed by Kneip et al.[37]

Appendix B

Homography from Motion

Given the geometry of the problem in Figure B.1, a point M in a plane P defined by plane equation $\mathbf{n}\mathbf{X}_1 = d$, were projected into the points m_1 and m_2 in image one and image two respectively[14]. We know the relation of the point m_1 in image one and point m_2 in image two were defined by the homography matrix $\mathbf{H}$.

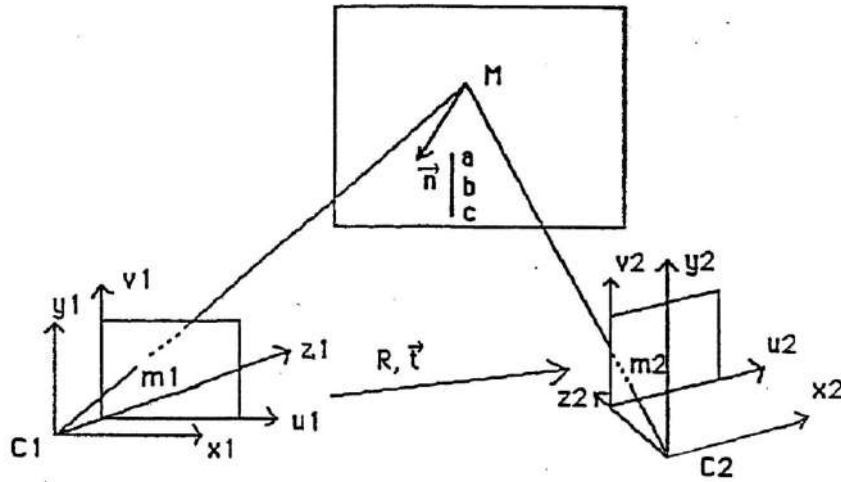


Figure B.1: Geometry of the problem.[14]

$$m_2 = {}^2\mathbf{H}_1 m_1 = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} m_1 \quad (\text{B.1})$$

Image one and image two were also related by the rigid transformation $(\mathbf{R}, \mathbf{t})$. Moreover, the homography matrix, ${}^2\mathbf{H}_1$ can be defined by the relation B.2

$${}^2\mathbf{H}_1 = d^2 \mathbf{R}_1 + {}^2\mathbf{t}_1 \mathbf{n}^\top \quad (\text{B.2})$$

The proof of B.2 was given by Faugeras[14] as follows. By definition, the coordinates $\mathbf{X}_1$ and $\mathbf{X}_2$ of M in the two canonical coordinate systems of the camera were related as

$$\mathbf{X}_2 = \mathbf{R}\mathbf{X}_1 + \mathbf{t} \quad (\text{B.3})$$

M was a point in the plane P defined by its normal, $\mathbf{n}$, and distance, d , to the canonical coordinate of the first camera. Therefore, the plane equation can be defined as

$$\mathbf{n}\mathbf{X}_1 = aX_1 + bY_1 + cZ_1 = d \quad (\text{B.4})$$

Combining Equation B.3 and Equation B.4 gave the relation

$$\mathbf{X}_2 = \left(\mathbf{R} + \frac{{}^2\mathbf{t}_1 \cdot \mathbf{n}_1^\top}{d_1} \right) \mathbf{X}_1 \quad (\text{B.5})$$

From [38], we knew that the homography between two images facing the same planar scene had a relation

$$\mathbf{X}_2 = {}^2\mathbf{H}_1 \mathbf{X}_1 \quad (\text{B.6})$$

Combining B.5 and B.6, we came to the conclusion that the homography matrix from image one to image two, ${}^2\mathbf{H}_1$, can be composed as follows

$${}^2\mathbf{H}_1 = {}^2\mathbf{R}_1 + \frac{{}^2\mathbf{t}_1 \cdot \mathbf{n}_1^\top}{d_1} \quad (\text{B.7})$$

Bibliography

- [1] Lin-Bo Luo et al. “Low-cost implementation of bird’s-eye view system for camera-on-vehicle”. In: *2010 Digest of Technical Papers International Conference on Consumer Electronics (ICCE)*. 2010, pp. 311–312. DOI: 10.1109/ICCE.2010.5418845.
- [2] Robert Krajewski et al. “The highD Dataset: A Drone Dataset of Naturalistic Vehicle Trajectories on German Highways for Validation of Highly Automated Driving Systems”. In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. 2018, pp. 2118–2125. DOI: 10.1109/ITSC.2018.8569552.
- [3] Dongfang Yang et al. “Top-view Trajectories: A Pedestrian Dataset of Vehicle-Crowd Interaction from Controlled Experiments and Crowded Campus”. In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. 2019, pp. 899–904. DOI: 10.1109/IVS.2019.8814092.
- [4] Z. Zhang. “A flexible new technique for camera calibration”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.11 (2000), pp. 1330–1334. DOI: 10.1109/34.888718.
- [5] Eric Marchand, Hideaki Uchiyama, and Fabien Spindler. “Pose Estimation for Augmented Reality: A Hands-On Survey”. In: *IEEE Transactions on Visualization and Computer Graphics* 22.12 (Dec. 2016), pp. 2633–2651. DOI: 10.1109/TVCG.2015.2513408. URL: <https://inria.hal.science/hal-01246370>.
- [6] I.E. Sutherland. “Three-dimensional data input by tablet”. In: *Proceedings of the IEEE* 62.4 (1974), pp. 453–461. DOI: 10.1109/PROC.1974.9449.
- [7] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Second. Cambridge University Press, ISBN: 0521540518, 2004.

- [8] Long Quan and Zhongdan Lan. “Linear N-point camera pose determination”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 21.8 (1999), pp. 774–780. DOI: 10.1109/34.784291.
- [9] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. “EPnP: An Accurate O(n) Solution to the PnP Problem”. In: *International Journal of Computer Vision* 81.2 (Feb. 2009), pp. 155–166. ISSN: 1573-1405. DOI: 10.1007/s11263-008-0152-6. URL: <https://doi.org/10.1007/s11263-008-0152-6>.
- [10] Kailin Tong, Zlatan Ajanovic, and Georg Stettinger. “Overview of Tools Supporting Planning for Automated Driving”. In: *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. 2020, pp. 1–8. DOI: 10.1109/ITSC45102.2020.9294512.
- [11] Alexey Dosovitskiy et al. “CARLA: An Open Urban Driving Simulator”. In: *Proceedings of the 1st Annual Conference on Robot Learning*. 2017, pp. 1–16.
- [12] Alexandre Robicquet et al. “Learning Social Etiquette: Human Trajectory Understanding In Crowded Scenes”. In: *Computer Vision – ECCV 2016*. Ed. by Bastian Leibe et al. Cham: Springer International Publishing, 2016, pp. 549–565. DOI: 10.1007/978-3-319-46484-8_33.
- [13] Wei Zhan et al. “INTERACTION Dataset: An INTERNATIONAL, Adversarial and Cooperative moTION Dataset in Interactive Driving Scenarios with Semantic Maps”. In: *arXiv:1910.03088 [cs, eess]* (Sept. 2019).
- [14] Olivier Faugeras and F. Lustman. “Motion and Structure from Motion in a Piecewise Planar Environment”. In: *International Journal of Pattern Recognition and Artificial Intelligence - IJPRAI* 02 (Sept. 1988). DOI: 10.1142/S0218001488000285.
- [15] Zefeng Lu et al. “A Lane Detection Method Based on a Ridge Detector and Regional G-RANSAC”. In: *Sensors* 19.18 (2019). ISSN: 1424-8220. DOI: 10.3390/s19184028. URL: <https://www.mdpi.com/1424-8220/19/18/4028>.
- [16] Hiroshi KANO et al. “Precise Top View Image Generation without Global Metric Information”. In: *IEICE Transactions on Information and Systems* E91.D.7 (2008), pp. 1893–1898. DOI: 10.1093/ietisy/e91-d.7.1893.
- [17] Ronald T Azuma. “A survey of augmented reality”. In: *Presence: teleoperators & virtual environments* 6.4 (1997), pp. 355–385.

- [18] Feng Zhou, Henry Duh, and Mark Billinghurst. “Trends in Augmented Reality Tracking, Interaction and Display: A Review of Ten Years of ISMAR”. In: *2008 7th IEEE/ACM International Symposium on Mixed and Augmented Reality 2* (Sept. 2008), pp. 193–202. DOI: 10.1109/ISMAR.2008.4637362.
- [19] A.J. Davison, W.W. Mayol, and D.W. Murray. “Real-time localization and mapping with wearable active vision”. In: *The Second IEEE and ACM International Symposium on Mixed and Augmented Reality, 2003. Proceedings.* 2003, pp. 18–27. DOI: 10.1109/ISMAR.2003.1240684.
- [20] E. Eade and T. Drummond. “Scalable Monocular SLAM”. In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’06)*. Vol. 1. 2006, pp. 469–476. DOI: 10.1109/CVPR.2006.263.
- [21] Jakob Engel, Thomas Schöps, and Daniel Cremers. “LSD-SLAM: Large-Scale Direct Monocular SLAM”. In: *Computer Vision – ECCV 2014*. Ed. by David Fleet et al. Cham: Springer International Publishing, 2014, pp. 834–849. ISBN: 978-3-319-10605-2.
- [22] H. Kato and M. Billinghurst. “Marker tracking and HMD calibration for a video-based augmented reality conferencing system”. In: *Proceedings 2nd IEEE and ACM International Workshop on Augmented Reality (IWAR’99)*. 1999, pp. 85–94. DOI: 10.1109/IWAR.1999.803809.
- [23] S. Malek et al. “Tracking chessboard corners using projective transformation for augmented reality”. In: *2011 International Conference on Communications, Computing and Control Applications (CCCA)*. 2011, pp. 1–6. DOI: 10.1109/CCCA.2011.6031196.
- [24] Michael Dille, Ben Grocholsky, and Stephen Nuske. “Persistent Visual Tracking and Accurate Geo-Location of Moving Ground Targets by Small Air Vehicles”. In: *Infotech@Aerospace 2011*. DOI: 10.2514/6.2011-1558. eprint: <https://arc.aiaa.org/doi/pdf/10.2514/6.2011-1558>. URL: <https://arc.aiaa.org/doi/abs/10.2514/6.2011-1558>.
- [25] Matthias Nieuwenhuisen et al. “Autonomous Navigation for Micro Aerial Vehicles in Complex GNSS-denied Environments”. In: *Journal of Intelligent and Robotic Systems* (Dec. 2016). DOI: 10.1007/s10846-015-0274-3.

- [26] R. Szeliski. *Computer Vision: Algorithms and Applications*. Texts in Computer Science. Springer International Publishing, 2022. ISBN: 9783030343729.
- [27] G. Bradski and A. Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media, 2008. ISBN: 9780596554040.
- [28] Peter Sturm and Srikumar Ramalingam. “A Generic Concept for Camera Calibration”. In: *8th European Conference on Computer Vision (ECCV '04)*. Ed. by Jirí Matas and Tomás Pajdla. Vol. 3022/2004. Lecture Notes in Computer Science (LNCS). Prague, Czech Republic: Springer-Verlag, May 2004, pp. 1–13. DOI: 10.1007/978-3-540-24671-8_1. URL: <https://inria.hal.science/inria-00524411>.
- [29] P.F. Sturm and S.J. Maybank. “On plane-based camera calibration: A general algorithm, singularities, applications”. In: *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*. Vol. 1. 1999, 432–437 Vol. 1. DOI: 10.1109/CVPR.1999.786974.
- [30] O. D. Faugeras, Q. T. Luong, and S. J. Maybank. “Camera self-calibration: Theory and experiments”. In: *Computer Vision — ECCV'92*. Ed. by G. Sandini. Berlin, Heidelberg: Springer Berlin Heidelberg, 1992, pp. 321–334. ISBN: 978-3-540-47069-4.
- [31] J. Heikkila and O. Silven. “A four-step camera calibration procedure with implicit image correction”. In: *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 1997, pp. 1106–1112. DOI: 10.1109/CVPR.1997.609468.
- [32] Joseph Redmon et al. “You Only Look Once: Unified, Real-Time Object Detection”. In: *CoRR* abs/1506.02640 (2015). URL: <http://arxiv.org/abs/1506.02640>.
- [33] Robert M. Haralick et al. “Review and analysis of solutions of the three point perspective pose estimation problem”. In: *International Journal of Computer Vision* 13 (1994), pp. 331–356.
- [34] Martin A. Fischler and Robert C. Bolles. “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography”. In: *Commun. ACM* 24 (1981), pp. 381–395.

- [35] George Terzakis and Manolis Lourakis. “A Consistently Fast and Globally Optimal Solution to the Perspective-n-Point Problem”. In: *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I*. Berlin, Heidelberg: Springer-Verlag, 2020, 478–494. ISBN: 978-3-030-58451-1. DOI: 10.1007/978-3-030-58452-8_28. URL: https://doi.org/10.1007/978-3-030-58452-8_28.
- [36] Yinqiang Zheng et al. “Revisiting the PnP Problem: A Fast, General and Optimal Solution”. In: *2013 IEEE International Conference on Computer Vision*. 2013, pp. 2344–2351. DOI: 10.1109/ICCV.2013.291.
- [37] Laurent Kneip, Paul Furgale, and Roland Siegwart. “Using multi-camera systems in robotics: Efficient solutions to the NPnP problem”. In: *2013 IEEE International Conference on Robotics and Automation*. 2013, pp. 3770–3776. DOI: 10.1109/ICRA.2013.6631107.
- [38] G. Simon, A.W. Fitzgibbon, and A. Zisserman. “Markerless tracking using planar structures in the scene”. In: *Proceedings IEEE and ACM International Symposium on Augmented Reality (ISAR 2000)*. 2000, pp. 120–128. DOI: 10.1109/ISAR.2000.880935.

List of Figures

1.1	Example of the highD dataset [2]	2
1.2	EZ Golf Cart as vehicle agent (left) and the markers on top of the car (right).[3]	3
1.3	CITR Dataset Setup: the pedestrians moved between two blue areas, the car moved between two green areas, and the drone stayed in the orange area.[3]	3
1.4	left: capturing pedestrian crosswalk, right: capturing a roundabout area.[3]	4
1.5	Experimental Setup of Human-Social Dataset[12]	4
1.6	Human-Social Dataset Example[12]	4
1.7	INTERACTION datasets example (a) urban merging, (b) highway ramp merging and lane change, (c)-(g) five roundabouts, and (h)-(j) unsignalised intersections, and (k) unprotected left turn at a signalised intersection.[13]	5
2.1	Carla Architecture.[11]	9
2.2	Visualization of Carla Modules.[11]	9
2.3	Top View of CARLA's Town 10 maps	10
2.4	Example View of Carla Town 10 Maps	11
2.5	CARLA uses the Unreal Engine coordinate system (left-handed system)[11].	11
2.6	Car Model	12
2.7	Image taken by car-mounted camera	13
2.8	Image taken by a 15 meters drone-mounted camera	13
2.9	Rotation effect on a drone-mounted camera positioned at 15 meters height.	13
2.10	Image taken by a 25 meters drone-mounted camera	14
2.11	Rotation effect on a drone-mounted camera positioned at 25 meters height.	14
2.12	Image taken by a 35 meters drone-mounted camera	14
2.13	Rotation effect on a drone-mounted camera positioned at 35 meters height.	15

2.14	CARLA's crosswalk dimension.	15
2.15	ELTE Car Camera Installation	16
2.16	ELTE Car Camera Configuration	16
2.17	Real World Experiment: Markers Position.	17
2.18	ELTE Experiment. Green box: ELTE Car, Red Box: Drone, Blue Box: Drone Pilot	17
2.19	Example Image Captured by Drone-Mounted Camera	18
2.20	Example of images taken by ELTE Car in experiment one.	18
2.21	Image Captured by Drone DJI Phantom 4 Pro V2.0	19
2.22	Example of images taken by ELTE Car in experiment two. (Notice a person gave a sign for time synchronisation in Dev0)	20
3.1	Sample of images for DJI Tello drone's camera calibration.	23
3.2	Sample of images for DJI Phantom 4 drone's camera calibration.	23
3.3	Pose estimation illustration taken from OpenCV documentation.	25
3.4	Planar homography from points lying on a common planar plane $\hat{n}_0 \cdot p + c_0 = 0$. [26]	26
3.5	The Proposed Method	27
3.6	Six points (red dots) were selected on the crosswalk as planar reference.	29
3.7	The coordinate system on the road plane.	30
3.8	Origin of the world coordinate system in image frame.	30
3.9	Pose estimation in the real world experiment.	31
3.10	Drone-mounted camera image (left) and its projective transformation result (right)	32
3.11	Comparison of the projective transformation result (left) with the image taken by the car-mounted camera (right)	33
3.12	Left: experiment one, right: experiment two	34
3.13	Drone pose at an altitude of 15 meters.	35
3.14	Drone pose at an altitude of 25 meters.	35
3.15	Drone pose at an altitude of 35 meters.	36
3.16	Chessboard Detection in the ELTE's Experiment 1.	36
3.17	Chessboard Detection in the ELTE's Experiment 2.	37
3.18	Comparison of the homography matrix derived from ${}^1\mathbf{T}_2$ and ${}^V\mathbf{T}_D$	38
3.19	Perspective transform of 15-meter-high drone camera's image.	39

3.20	Perspective transform of 25-meter-high drone camera's image.	40
3.21	Perspective transform of 35-meter-high drone camera's image.	40
3.22	Perspective transform of ELTE Experiment 1.	41
3.23	Perspective transform of ELTE Experiment 2.	41
3.24	15-meter high drone image with rotation.	43
3.25	25-meter high drone image with rotation.	43
3.26	35-meter high drone image with rotation.	43
3.27	Image taken by the car-mounted camera.	45
3.28	Red box showed the location of an identified car.	45
3.29	Perspective transformation showed the car's location (green arrow).	46
3.30	Annotated images.	46
3.31	Transformed annotated images.	47
3.32	Drawing bounding box using location from transformed images.	47
B.1	Geometry of the problem.[14]	54

List of Algorithms

1	Alignment of Car-Mounted and Drone-Mounted Cameras	28
2	Updating Homography	28