CHALMERS
UNIVERSITY OF TECHNOLOGY

Surround-view for Articulated Vehicles

Sensor fusion and image processing for articulated vehicles in a surround-view system

Master's thesis in Systems, Controls & Mechatronics

Hubert Hermodsson
Albin Svenske

DEPARTMENT OF MECHANICS AND MARITIME SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026
www.chalmers.se

Master's thesis 2026

Surround-view for Articulated Vehicles

Sensor fusion and image processing for articulated vehicles in a surround-view system

HUBERT HERMODSSON
ALBIN SVENSKE

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mechanics and Maritime Sciences
Vehicle Engineering and Autonomous Systems
Chalmers University of Technology
Gothenburg, Sweden 2026

Surround-view for Articulated Vehicles
Hubert Hermodsson, Albin Svenske

Supervisor: Björn Johnsson, CPAC Systems AB
Examiner: Peter Forsberg, Department of Mechanics and Maritime Sciences

Master's Thesis 2026
Department of Mechanics and Maritime Sciences
Vehicle Engineering and Autonomous Systems
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Wheel loader with illustrated camera field-of-views.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2026

Surround-view for Articulated Vehicles
HUBERT HERMODSSON, ALBIN SVENSKE
Department of Mechanics and Maritime Sciences
Chalmers University of Technology

Abstract

Bird's-eye-view surround-view systems are widely used for driver assistance in rigid vehicles operating on flat terrain but are poorly suited for articulated vehicles and uneven terrain. This thesis investigates the feasibility of extending surround-view systems to articulated vehicles in uneven terrain using sensor fusion and image processing.
A surround-view pipeline is developed that combines camera-based bird's-eye-view generation with inertial measurement data from Inertial Measurement Units (IMUs). A key contribution is the introduction of independently tilted ground planes for each vehicle frame, enabling camera projections to adapt dynamically to vehicle articulation and terrain orientation. Inertial Measurement Unit-camera fusion is applied for orientation estimation, while image undistortion and adaptive stitching are used to improve visual consistency.

The approach is evaluated in simulation and through real-world experiments using relatively low-cost cameras and commercially available Inertial Measurement Units. The results demonstrate improved geometric consistency on uneven terrain compared to conventional flat-plane bird's-eye-view methods and confirm the feasibility of the system as a proof of concept. The work prioritizes computational efficiency and comprehensibility over dense three-dimensional reconstruction, indicating that IMU-camera fusion with per-frame tilted ground planes is a viable and efficient approach for surround-view systems for articulated vehicles.

Keywords: Surround-view, Articulated vehicles, Image stitching, Vehicle safety, Image processing, IMU, Bird's-eye-view.

Acknowledgements

First and foremost, we would like to thank our examiner, Peter Forsberg, for his drive and assistance during the planning, process, and thesis writing. It was the needed boost to finish the thesis. We would also like to extend a special thanks to our advisor at CPAC, Björn Johnsson, for his guidance and assistance throughout the entire project.

Additionally, we would also like to thank CPAC for the opportunity to carry out our thesis in collaboration with them and the resources offered to us, including workspace, equipment, coffee, and Wednesday fika.

Hubert Hermodsson & Albin Svenske, Gothenburg, May 2026

Thesis advisor: Björn Johnsson, CPAC Systems AB
Thesis examiner: Peter Forsberg, Mechanics and Maritime Sciences

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

ADAS	Advanced Driver Assistance Systems
ArUco	Augmented Reality University of Córdoba
BEV	Bird's-Eye-View
CAN	Controller Area Network
ChArUco	Chessboard + ArUco
CPAC	CPAC Systems AB
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
FOV	Field Of View
FPS	Frames Per Second
GPU	Graphics Processing Unit
IMU	Inertial Measurement Unit
IP	Internet Protocol
IPM	Inverse Perspective Mapping
KNN	k-Nearest Neighbors
LiDAR	Light Detection And Ranging
MIMC	Multi-IMU, Multi-Camera
OpenCV	Open Computer Vision Library
ORB	Oriented FAST and Rotated BRIEF
RANSAC	Random Sample Consensus
SIFT	Scale-Invariant Feature Transform
std	Standard Deviation
VINS	Visual Inertial Navigation System

Nomenclature

Below is the nomenclature of indices, sets, parameters, and variables that have been used throughout this thesis.

Indices

k	Step identifier for a Kalman update step

Parameters

$d_{\varphi,\theta}$	Damping for roll and pitch in filter motion model [1/s]
d_{ψ}	Damping for yaw in filter motion model [1/s]
$s_{\varphi,\theta}$	Stiffness for roll and pitch in filter motion model [$1/s^2$]
$\mathbf{K}$	Camera intrinsic matrix mapping camera coordinates to pixel coordinates
$\mathbf{D}$	Array of distortion coefficients representing the distortion from a camera lens
$\mathbf{T}_{\text{world}}^{\text{pix}}$	Homogeneous metric-to-pixel transformation from world-plane coordinates to BEV image coordinates

Sets

$SE(3)$	Special Euclidean group in 3D, representing 3×3 rotation matrices
$SO(2)$	Special Orthogonal group in 3D, representing rigid body transformations

Variables

φ	Roll angle of a body, around the forward pointing axis [deg]
θ	Pitch angle of a body, around the left pointing axis [deg]
ψ	Yaw angle of a body, around the upwards pointing axis [deg]
$\dot{\varphi}$	Roll rate of a body, around the forward pointing axis [deg/s]
$\dot{\theta}$	Pitch rate of a body, around the left pointing axis [deg/s]
$\dot{\psi}$	Yaw rate of a body, around the upwards pointing axis [deg/s]
b_ψ	Drift bias in the yaw dimension as a filter state [deg]
$\mathbf{R}_a^b$	Rotation matrix that represents how a frame b is rotated with respect to a frame a
$\mathbf{t}_a^b$	translation vector representing how a frame b is translated with respect to frame a [m]
$\mathbf{H}$	Planar homography matrix
$\mathbf{H}_a^b$	planar projective transformation mapping points from frame a into frame b
$\mathbf{p}_i$	World position of camera i [m]
$\mathbf{T}_i$	Rigid-body transformation of camera i from the world frame, consisting of rotation and translation [[SE(3)]]
$\mathbf{f}_i$	Optical forward direction of camera i expressed in world coordinates

Contents

List of Figures

List of Tables

1
Introduction

Today, an increasing number of vehicles are implementing a Bird's-Eye-View (BEV) surround system to better understand and recognize obstacles in their surrounding environment. This helps primarily with parking and is, in general, mostly used in personal vehicles. The market for passenger cars and for commercial vehicles is increasing and indicates continued growth as more and more manufacturers request these kinds of systems [6]. A surround-view is a visual tool created to help a driver of a vehicle better understand and monitor its nearby environment and close-by features. The view is created by stitching together real-time images from multiple cameras around the vehicle in order to create a 360-degree BEV, much like shown in Figure 1.1. The feature is especially useful for parking or driving in narrow environments, where visibility can be limited [7]. This is one critical feature for developing an Advanced Driver Assistance System (ADAS). However, most of these existing approaches are simple, adapted mostly for even terrain and for smaller rigid vehicles.

Figure 1.1: Bird's-eye-view of a small rigid vehicle in a parking lot by a drone. Representation of what is referred to as a BEV [1].

For most of the other cases with, for example, trucks, buses and other diverse machinery, these solutions do not exist to the same extent and there is a lot of development yet to be done. As many of these types of vehicles also navigate in tight spaces and often with more expensive and heavy equipment, it could have a significant impact in terms of safety and efficiency if the technology can be implemented here as well. To achieve this, multiple sensors can be utilized, for example, cameras and Inertial Measurement Units (IMUs) which are becoming increasingly common and affordable [8]. Therefore, they are an interesting combination to look into for developing an advanced surround-view system.

In collaboration with CPAC Systems AB, hereafter abbreviated as CPAC, this master's thesis has investigated a surround-view implementation that generates a dynamic surround camera view around an articulated vehicle using techniques such as sensor fusion, image stitching and other image processing techniques with a goal of being a proof of concept for future implementation in CPAC products.

1.1 Purpose

The purpose of this master's thesis is to investigate an implementation of a bird's-eye surround-view, hereafter referred to as a surround-view, around an articulated vehicle. It will investigate and develop a method for combining camera data together with IMUs to estimate the pose of each vehicle frame. This approach will be used in order to improve the online extrinsic calibration for each camera, as to allow for dynamic image stitching used in the projection and creation of a representative surround-view, while aiming to be more computationally efficient and robust than previous methods that only use cameras.

This project will additionally investigate how, with available resources, the driver's perception and understanding of the environment can be improved when it comes to uneven terrain and situations where visibility is limited from traditional viewpoints.

1.2 Related work

Regarding surround-view systems, there is somewhat of a gap between the amount of academic research and publications on the area compared to the widespread implementation in industry as written by Kumar et al. [9]. The authors attribute this to a lack of relevant datasets of fisheye cameras and 360° coverage. However, some research on relevant systems does exist, some of them are put forward in the following section.

As stated in Section 1, existing implementations of the system described are adapted and exist in production for 1-body vehicles with a rigid frame. Kumar et al. writes about the existing technology in a review. The systems are used mostly in personal vehicles for assisted parking, adapted for low-speed urban driving and a good way for blind spot monitoring.

The authors also touch upon the usual sensor setup. Most systems today utilize 4 fisheye cameras evenly spaced out and rigidly mounted on the vehicle frame to get a 360° near-field view of its surroundings. The implementation is using homography-based methods where a bird's-eye view is generated and projected on a ground plane projection surface.

Regarding choice of sensors for this kind of system, one source [10] written by Marko et al. discusses the possibilities of accurately and robustly estimating the rotation of

a vehicle frame and its ground plane by combining IMU-data with camera-odometry through a Kalman filter. This paper performs this to estimate the tilt of a single vehicle frame in order to give better information of the vehicle state to the driver and vehicle technology. As these are the same measurements needed for this project, it is highly relevant looking at the conclusions from this study, which has shown that accurate prediction of the ground-plane normal is important for several perception tasks, including bird's-eye-view perception through inverse perspective mapping (IPM). Their results ultimately show that IMU-camera fusion significantly improves robustness when road surfaces exhibit changes in slope or elevation.

Regarding sensor fusion of the camera-IMU combination, also referred to as Visual Inertial Navigation System (VINS) and mentioned in the previous paragraph, a review in 2019 [11] mapped out where research within this area stood at the time. It established that the field had generated a lot of traction in the research community and that methods surrounding it have proven effective and support a wide set of applications. Most of the research has focused on visual odometry, i.e., tracking one object over time based on information from its camera.

An additional paper describes a Multi-IMU, Multi-Camera (MIMC) algorithm[8] which use an arbitrary number of sensors, cameras and IMUs together with a Kalman filter for 3D motion tracking. Even though the research and testing were performed on a rigid frame and not with the intention of measuring the relative angle between bodies, the research is still interesting for the use case of this thesis. They support this algorithm with remarks on how this kind of sensor combination is gaining traction in popularity and proving effective in precisely the environments where this thesis aims at solving problems in.

Projects with a similar goal as this thesis [12], [13] have generally only utilized the cameras themselves as position sources with the help of feature matching and often neural networks for obtaining position. However, this approach can lack in performance when the environment is low-texture or otherwise challenging, which is the case in visual odometry as mentioned by Wang et al. [14]. Minervini et al. mention the same [15] and suggest multiple methods for countering this, one of them being to implement IMUs.

In light of the above, a number of papers describe the process of using the combination of camera and IMU for positioning of bodies in different scenarios and list the advantages as robustness, accuracy and affordability. These reasons together give a clear indication that using IMUs and cameras together could be a promising foundation to help solve the problems defined in this project.

In regards to creating the surround-view itself, there are previous research papers that describes the process of creating a computationally efficient surround-view around an articulated vehicle, they aim to solve the dynamic image stitching by fusing together photometric data with a joint angle sensor as discussed by Li et al. [16]. It faced the challenge of handling large distortion errors as a consequence of

alternating angles of the vehicle frames and accumulative error of the angle sensor. This was achieved by performing the online photometric-error minimization.

Dong et al. presents a more recent approach for articulated tractor-trailer vehicles, where a stitched 2D panorama around the vehicle is mapped onto a novel 3D drop model [12]. The stitching is based on estimating the relative poses of the cameras intersecting each other through feature matching, during articulation of the vehicle.

Importantly, both papers by Li et al. [16] and Dong et al. only test their implementations in an environment where the ground is flat, an assumption that does not always hold true in all real-world scenarios, as for example when the tractor-trailer vehicle traverses over a hill. As a consequence, projection accuracy and stitching consistency could degrade when existing methods are applied outside the idealized flat-ground setting.

1.3 Research questions

The main research questions investigated throughout this master's thesis are the following:

- How can IMUs and cameras be utilized in order to produce accurate rotational data to use for creating a representative surround-view for articulated vehicles?
- How can multi-dimensional orientation data be used in order to better represent uneven terrain for the operator in a more varied and unpredictable environment?
- What image processing techniques can be used to fuse images together in order to create a realistic and usable surround-view for the user driving an articulated vehicle?

1.4 Scope and limitations

This project will investigate the possibility of a bird's-eye surround-view for articulated vehicles with two frames (tractor- 1-trailer systems), as this encapsulates requests from the market and incorporates the majority of popular machinery. This does not include dollies which are common in some trailer setups that is essentially creating a 3-body system making the setup more intricate and not applicable for this project.

Only combinations of cameras and IMUs will be investigated. This is due to cost as implementing for example Light Detection and Ranging (LiDAR) sensors for depth perception and orientation estimation is expensive. The request from CPAC is a cheaper and more modular system, and it is therefore more appropriate using cheaper cameras and already in-house built IMUs.

The setup investigated will only take into account the case of 6 cameras according to the rough setup in Figure 1.2, as to place the effort in investigating the stitching and sensor fusion techniques rather than the optimal number of cameras and their respective placement.

Figure 1.2: Sensor setup of the 2-body vehicle, the basis of this project.

Additionally, the cameras' extrinsic and intrinsic calibration is not a central part of the project, it will thus not be prioritized. This results in several consequences.

- The cameras' extrinsics will not be algorithmically calibrated; they will be measured manually, both in position and rotation. This might result in poorer alignment in the stitching as well as some performance loss in orientation estimation as it relies on accurate extrinsic calibration.
- The cameras themselves must be intrinsically calibrated in order to best obtain the exact amount of distortion. This will not be a focus area either, however, a simple calibration will be carried out with a basic script and the OpenCV library. All of the Internet Protocol cameras (IP-cameras) are assumed to carry the same intrinsic characteristics, as they are of the same product line. An intrinsic calibration will therefore only be carried out for one of the cameras and then assumed to be the same for all the other cameras.
- At system startup, the yaw angle between the vehicle frames is assumed to be zero as no start-up-calibration algorithm will be developed to calibrate the initial relative angle between the IMUs. It is therefore important to make sure that the machine is perfectly straight before initializing the system, ensuring that the correct data is sent to the system. Otherwise, a constant offset will be injected.

1.5 Original contribution

To the best of the authors' knowledge, no previous papers have chosen to use a multiple-camera and IMU combination in order to obtain robust and high-frequency data for building a surround-view system for articulated vehicles.

Using a dynamic ground plane based on orientation data has been documented previously for surround-view systems in rigid vehicle settings, however using it for an articulated vehicle where their ground planes may vary significantly, has not, to date, been documented in existing literature.

2
Theory

This chapter will present the theoretical principles and areas necessary to understand the later presented methodology of this thesis. Section 2.1 defines articulated vehicles. Section 2.2 outlines the theory behind surround-views and their core components. Section 2.3 introduces the scripting functions used from OpenCV. Section 2.4 describes the fusion techniques applied. Section 2.5 presents the rotation matrix conversion equations, and Section 2.6 lists the hardware used.

2.1 Articulated vehicles

An articulated vehicle is a multi-body road vehicle with a permanent or semi-permanent coupling. There are multiple different kinds of articulated vehicles, some examples include articulated buses, tractor-semitrailer combinations and specialized multi-carriage industrial vehicles. In Figure 2.1, an example of one type of articulated vehicle is shown. A defining characteristic of this vehicle family is that their pivot joint introduces kinematic behavior, which is a fundamental difference from rigid vehicles where there is only one body segment.

Figure 2.1: An example of an articulated vehicle, a Caterpillar 740 Ejector articulated haul truck driving up a hill [2].

The coupling is acting as a large pivot joint, also referred to as an articulation joint. The coupling allows the different body segments to rotate relative to one another,

most commonly around the vertical axis when seen from above, but in real scenarios it also rotates around its lateral (pitch) and longitudinal (roll) axes since the terrain is often irregular, because of changing suspension dynamics and of mechanical play. The different articulation joints vary in their mechanical design, dependent on the vehicle type and the intended environment to operate in. Figure 2.2 shows an articulation joint mechanism used in a wheel loader, where the vehicle is divided into two rigid body segments connected by a centrally located pivot.

(a) A wheel loader with an articulation joint, shown from the side.

(b) Side view showing the hitch joint placement.

Figure 2.2: Wheel loader with an articulation joint [3].

In contrast, Figure 2.3 shows a bus and the placement of its articulation joint. While the mechanical design differs from that of the wheel loader, the underlying kinematic principles remains the same, two rigid segments connected through an articulation joint that allows relative motion between them.

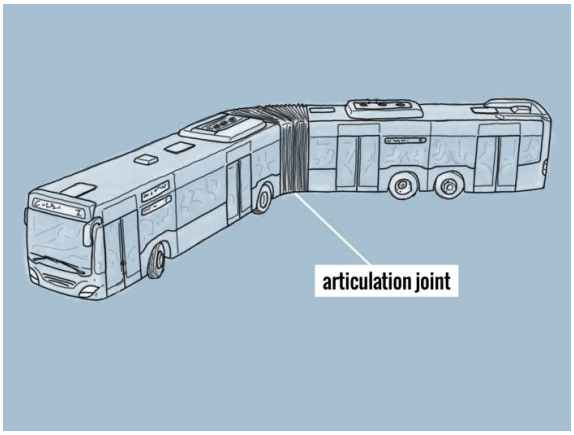


Figure 2.3: Illustration of an articulated bus and its articulation joint placement [4].

2.2 Surround-view systems

Surround-view systems, also known as around-view monitors or bird's eye-view (BEV) systems are a class of driver-assistance technologies designed to provide a

360° top-down visualization around the ego-vehicle. Their primary objectives are to reduce blind spots, enhance situational awareness, parking and navigating tight spaces.

To obtain a well-represented BEV around the vehicle certain processing steps are required. The camera intrinsic calibration finds the internal imaging parameters to relate the pixel coordinates to the camera coordinate system, the extrinsic calibration defines the spatial relationship between each camera, the vehicle frames and the world coordinate system. By using these parameters, the planar homographies can be computed to map the image points onto a common ground plane, which enables inverse perspective mapping as described in Subsection 2.2.3. When using multiple cameras to get a full 360° projection of the surroundings, the individual projected image views must be spatially aligned and then stitched together. To achieve seamless seam stitching, different blending methods can be used in order to reduce these effects.

2.2.1 Intrinsic calibration

Intrinsic calibration defines the parameters focal length, principal point and distortion which are often obtained by using a checkerboard-based algorithm using for example AruCo or ChAruCo markers. These are mature methods and widely used and supported in automotive engineering. The estimated parameters help to describe the intrinsic parameters of a camera model as seen in Equation 2.1 and 2.2, containing a camera matrix and distortion coefficients. It is worth noting that the distortion coefficients vary depending on the type of camera model, either a classic pinhole or a fisheye camera.

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

$$\mathbf{D}_{\text{pinhole}} = \begin{bmatrix} k_1 & k_2 & p_1 & p_2 & k_3 \end{bmatrix}, \qquad \mathbf{D}_{\text{fisheye}} = \begin{bmatrix} k_1 & k_2 & k_3 & k_4 \end{bmatrix} \tag{2.2}$$

The motivation for estimating these parameters is to accurately map pixels to real-world points by accounting for lens distortion. This is especially relevant for ultra-wide cameras, which introduce significant distortion that heavily impacts the results of mapping pixels to real-world locations. This is exemplified in Figure 2.4a where the clearest example of the effect is how the road is curved in a way that can be described by the cameras' intrinsics. In the rectified image shown in Figure 2.4b, the lines are straight and objects appear undistorted which is needed in order to make pixel-based calculations.

(a) Example of an image carrying effects of a wide-angle lens [17].

(b) Representation of a rectified image of the Eiffel tower [18].

Figure 2.4: Example of a wide lens and rectified images.

2.2.2 Extrinsic calibration

Extrinsic calibration describes how each camera is positioned and oriented relative to the coordinate frame it is attached to. Getting accurate extrinsic calibration is important, since small pose errors lead to visible misalignment in the stitched projected BEV image. Today, there are modern approaches using multi-camera optimization loops to refine the extrinsic calibration step. The extrinsic parameters can be defined in a combined 3x4 matrix $[\,\mathbf{R} \mid \mathbf{t}\,]$, it includes a 3x3 rotation matrix $\mathbf{R}$ and a 3x1 translation vector $\mathbf{t}$.

$$[\,\mathbf{R} \mid \mathbf{t}\,] = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \tag{2.3}$$

The cameras should, by standard computer vision practice, have their optical axes aligned with the z-axis of the camera coordinate system as described by Hartley et al. [19]. An illustration of the camera coordinate system with respect to the world coordinate system can be viewed in Figure 2.5.

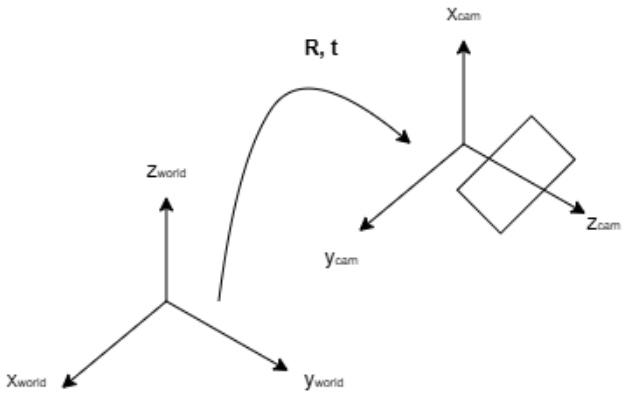


Figure 2.5: From world to the camera's coordinate system.

2.2.3 Estimation of a plane-to-image homography

A homography is a geometric transformation that preserves straight lines. A homography maps points between coordinate systems or images. Its transformation is shown in Equation 2.4 and represented by a 3×3 matrix [19]. Here, the subscripts indicate the mapping direction, where $H_{\mathrm{A}}^{\mathrm{B}}$ performs the transformation from $A \rightarrow B$.

$$\mathbf{x}' = H_{\mathrm{A}}^{\mathrm{B}}\mathbf{x} \tag{2.4}$$

Within the area of BEV generation, the objective is to find the homography taking the camera coordinates into a chosen ground planar surface. Equation 2.5 is derived by using the camera intrinsics and extrinsics from Equations 2.1 and 2.3. This equation describes the relation of projecting points from the real world into pixels in the image coordinate system; it is called the standard pinhole camera projection equation as described by Hartley et al [19].

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \mathbf{K}[\,\mathbf{R} \mid \mathbf{t}\,] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.5}$$

However, Equation 2.5 is not yet a homography since a homography is a 3x3 mapping between two planes, not between 3D space and the image. By convention, the plane is defined by setting $Z = 0$, which reduces the projection to a homography as shown in Equation 2.6.

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \mathbf{K} \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = H_{\text{plane}}^{\text{img}} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \tag{2.6}$$

The symbol λ represents the non-zero projective scale factor and arises from using homogeneous coordinates. In practice, this factor is eliminated through normalization and therefore λ is not explicitly tracked in later computations.

To instead get the projective transformation from the 2D perspective camera image to the BEV representation, inverse perspective mapping (IPM) converts the image and reduces perspective distortion. This is done by taking the inverse of the homography matrix in Equation 2.6, Equation 2.7 presents this homography inversion. In Figure 2.6, an illustrative example of the inversion is shown with the final projection; the 3D world is projected onto a 2D image showing a planar BEV.

$$H_{\text{img}}^{\text{plane}} = \left(H_{\text{plane}}^{\text{img}}\right)^{-1} \tag{2.7}$$

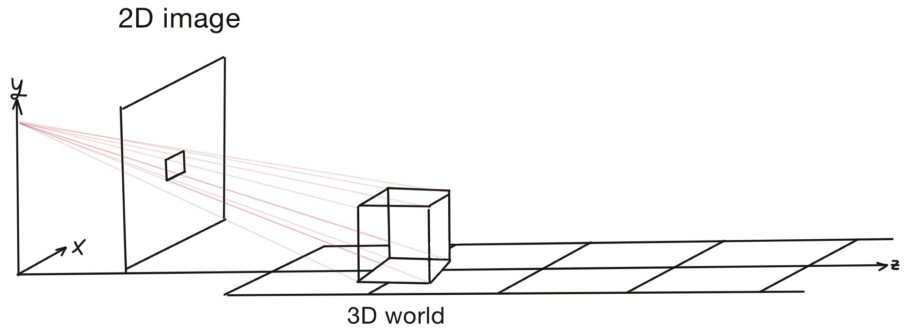


Figure 2.6: Projection from the image to the ground plane homography by using inverse perspective mapping.

2.2.4 Stitching and blending

When creating a surround-view system, multiple cameras are utilized for full coverage around the vehicle, resulting in an overlapping region between two adjacent cameras after they are projected to the ground plane. Stitching refers to the process of aligning these individual BEV projections into a common coordinate system using the estimated calibration parameters and homographies as described in Sections 2.2.1, 2.2.2 and 2.2.3. This is done to ensure consistency in the overlapping regions.

Despite accurate calibration, geometric inaccuracies and photometric differences often remain between cameras. These effects may arise from calibration errors, lens distortion effects and variation in viewing angle. Therefore blending techniques are performed to reduce the pixel errors from the previously mentioned effects, where they could have caused visible seams, differences in lighting intensity and other color mismatches. Common blending approaches include hard seam selection, linear feathering and multi-band blending. Each of these has its own trade-offs between computational cost and the visual quality they offer. In surround views, blending is typically used in the overlapping regions between two projections along the seam to smooth the transition.

2.3 OpenCV

OpenCV is a library of programming functions mainly used for real-time computer vision [20]. The name is an abbreviation for Open Source Computer Vision Library. The library is cross-platform and licensed as free and open-source software. Within the context of this master's thesis, OpenCV will be used in Python scripts. The functions used from OpenCV during this project will be presented in this subsection.

2.3.1 cv.findEssentialMat()

For finding the essential matrix **E** which describes the relation between two perspectives of the same point, this function can be used by inputting matched points

from the first and second perspective as well as camera intrinsic parameters and RANSAC settings.

2.3.2 cv.recoverPose()

The function `cv.recoverPose()` is usually used as a subsequent step after calculating the essential matrix. This function is from the essential matrix, recovering a rotation matrix and translation vector describing the rotation and distance between the two perspectives. This function also returns the inliers, which is a list of the matched points used in creating the essential matrix that supports the estimated transformation.

2.3.3 cv.warpPerspective()

For a BEV projection, the function `cv.warpPerspective` can be used to warp images to their correct dimensions. The function takes an image, a 3×3 transformation matrix and a specified size of the output image as input, and produces a warped image in the target coordinate frame.

2.3.4 cv.fisheye.initUndistortRectifyMap()

This function helps to compensate for radial and tangential distortion from wide-angle and fisheye lenses. It computes undistortion and rectification maps for image transform, which transforms distorted image coordinates into an undistorted rectified image. The inputs are the camera intrinsic matrix and distortion coefficients as described in Subsection 2.2.1 and the undistorted image size.

2.3.5 cv.distanceTransform()

The function `cv.distanceTransform` computes the distance from each pixel to the nearest zero-valued pixel. It inputs the source image, the distance type and the size of the mask. The output is a distance map where higher values correspond to pixels farther from the boundaries of the mask.

2.4 Sensor fusion

Sensor fusion is a wide scientific and industrial field which means fusing multiple sensors, often of different varieties to estimate relevant model states and information relevant for either a system or an operator. A common way of performing sensor fusion is through different kinds of filters where Kalman filters, as described in Section 2.4.1 are one of the most common.

2.4.1 Kalman filter

Kalman filters are frequently used within engineering and system design, mainly in areas such as guidance, navigation and control of vehicles[21], mainly within sensor fusion. A regular Kalman filter is used to estimate linear motion and observation models with Gaussian noise by alternating between two steps: prediction and update. The prediction steps use the previous time state estimation to predict what the current time state should be according to the motion model and some added noise. In the update step, the most recent measurements act as a correction of the just predicted state by looking at 1. What the measurement is, 2. What noise can be expected in this measurement?, 3. Is this measurement reasonable considering earlier motion? (innovation). The Kalman update equations are given by Equation 2.8:

$$
\begin{aligned}
&\textbf{Prediction step:} \\
\hat{x}_{k|k-1} &= F_k \hat{x}_{k-1|k-1} + B_k u_k \\
P_{k|k-1} &= F_k P_{k-1|k-1} F_k^{\mathrm{T}} + Q_k \\
&\textbf{Update step:} \\
\tilde{y}_k &= z_k - H_k \hat{x}_{k|k-1} \\
S_k &= H_k P_{k|k-1} H_k^{\mathrm{T}} + R_k \\
K_k &= P_{k|k-1} H_k^{\mathrm{T}} S_k^{-1} \\
\hat{x}_{k|k} &= \hat{x}_{k|k-1} + K_k \tilde{y}_k \\
P_{k|k} &= (I - K_k H_k) P_{k|k-1}
\end{aligned}
\tag{2.8}
$$

2.4.2 VINS

Visual-Inertial Navigation Systems (VINS) refer to navigation systems that perform visual odometry using cameras and refine the results with onboard IMU data.

This combination of sensors has a few advantages, mainly that the sensors complement each other well. The IMUs perform well during rapid orientation changes and can keep up with the fast motion of an object. The IMUs work quite well in the long run as well due to the accelerometer, but can suffer some drift in yaw due to the lack of gravitational forces. While cameras struggle with rapid motion, they can correct long-term IMU drift. This complementary relationship theoretically yields accurate data across both short-term dynamics and long-term trajectories. The VINS combination also offers the opportunity of slightly higher performance as reliance on only heavy camera-based algorithms is relaxed.

2.5 Euler angle to rotation matrix

Euler angles use three sequential rotations about predefined axes. In this work, a ZYX Euler angle convention is used, corresponding to a rotation about the z-axis (yaw), followed by a rotation about the y-axis (pitch) and finally a rotation about

the x-axis (roll). The ZYX convention used here corresponds to intrinsic rotations, meaning that each rotation is applied about the axis of the rotating coordinate frame.

The elemental rotation matrices for roll, pitch and yaw are defined as:

$$\mathbf{R}_x(\varphi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi & -\sin\varphi \\ 0 & \sin\varphi & \cos\varphi \end{bmatrix} \tag{2.9}$$

$$\mathbf{R}_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{2.10}$$

$$\mathbf{R}_z(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.11}$$

The combined rotation matrix describing the orientation of a frame with respect to another is obtained by composing the elemental rotations in ZYX order:

$$\mathbf{R}_{zyx} = \mathbf{R}_z(\psi)\,\mathbf{R}_y(\theta)\,\mathbf{R}_x(\varphi) \tag{2.12}$$

2.6 Hardware

This subsection will showcase the specific hardware used for the real-world testing during the project. This includes the IMUs in Subsection 2.6.1 and the IP-cameras in Subsection 2.6.2.

2.6.1 CPAC IMUs

As stated from the data sheet on CPAC's website, CPAC's IMU offers accelerometer, gyroscope and magnetometer to provide positioning for different vehicle types [22]. The IMU is constructed with an internal filter that fuses gyroscope and accelerometer readings to give a rotation of the IMU in relation to a predefined rotation. By using Earth's gravitational acceleration, it is possible to obtain absolute angles in two dimensions. Many IMUs also incorporate a magnetometer so that the yaw angle can be accurately calculated without accumulating drift. However, due to the fact that the use case for this IMU is to be mounted on heavy construction equipment, which consists of several metric tons of steel, the magnetometer is especially subject to interference and disturbance. This reduces the effectiveness of the magnetometer and hinders its implementation in a high-accuracy filter.

Figure 2.7: A CPAC IMU, which is the IMU used throughout this project [5].

To communicate with the IMU and to be able to obtain the different kinds of data produced by the sensor, one can open up communication to it via a vehicle communication bus standard SAE, also referred to as SAE J1939[22], which is based on the CAN standard and a CAN-dongle. Via this protocol, it is possible to directly read for example angular data in roll, pitch and yaw with a 20 Hz sampling rate from the bus.

2.6.2 STONKAM IP-cameras

The cameras used for real-world testing are wide-angle IP-cameras from STONKAM [23], which have a diagonal 150° FOV, classifying it as an ultra-wide lens. The cameras deliver 1080p resolution via an Ethernet connection, allowing live communication between a computer and the cameras.

Figure 2.8: The STONKAM IP-cameras used throughout the project.

3

Methods

Developing the BEV system consisted of multiple steps. Firstly, a literature review was conducted on possible existing solutions and techniques used for similar issues. Thereafter, an initial system pipeline was built that could be rapidly developed end-to-end, first with simulated camera data and subsequently with data from actual wide-angle lens cameras in a real-world environment. Finally, the system was evaluated further refined and optimized.

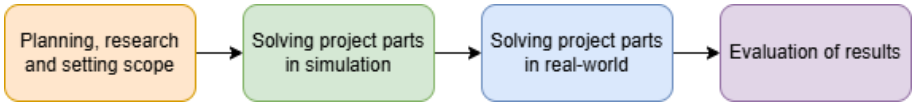


Figure 3.1: Rough timeline of the project development.

The methods chapter outlines the assumed configuration of the vehicle platform, the sensors it has incorporated and their approximate relative placement to each other. This is followed by a description of how everything was set up in simulation and in real-world tests. Then, the system design is detailed, covering orientation estimation and the image processing leading to the bird's-eye-view projection. Finally, the process for evaluation is presented.

3.1 Vehicle model

A general articulated vehicle platform was established as a consistent baseline for the system development. The configuration was intended to capture the main characteristics of an articulated vehicle as well as be equipped with an appropriate setup of sensors, both in numbers and type.

A tractor-trailer setup where the world rotation of each vehicle frame could be measured, as seen in Figure 3.2 was selected as it, in a general sense represents the characteristics of most articulated vehicles relevant to the scope of this project.

Figure 3.2: Image of the "tractor-trailer" configuration used for testing and development in the game engine Unity.

The vehicle is simplified with a hitch that allows free movement around three rotational axes, enabling operation on uneven terrain. This 3D-articulation is a key characteristic of construction, forestry, and similar machines.

When the vehicle is turning, the articulation angle represents how the trailer is rotated in relation to the tractor, i.e., the difference in world rotation between the frames. The articulation angle is a 3D-vector containing individual rotation in each dimension (roll, pitch and yaw). In the case of this system, a positive articulation in for example yaw, looks as in Figure 3.3.

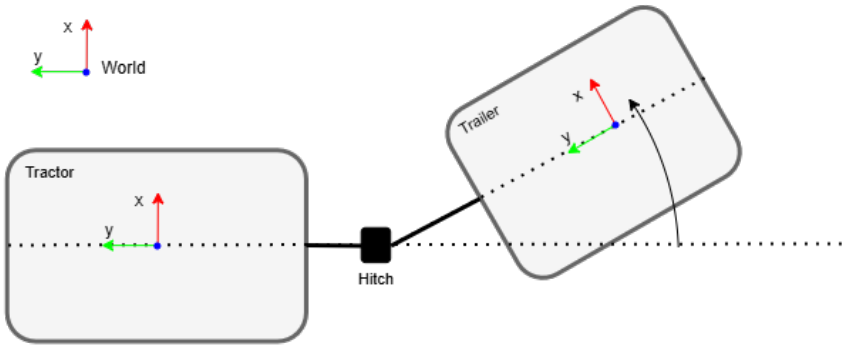


Figure 3.3: Example of positive articulation around the z-axis where $\psi \approx 30°$.

When building the surround-view, the assumption that the operator is sitting in the front part of the vehicle (tractor) is used. Following this assumption, it was reasonable to keep the tractor frame fixed in yaw during projection to match the cabin orientation, while preserving pitch and roll tilt. The trailer is thereafter ro-

tated around the hitch with the relative rotation as presented in Equations 3.1, 3.2 and 3.3.

$$\begin{bmatrix} \varphi_{rel} \\ \theta_{rel} \\ \psi_{rel} \end{bmatrix} = \begin{bmatrix} \varphi_{w \to trailer} \\ \theta_{w \to trailer} \\ \psi_{w \to trailer} \end{bmatrix} - \begin{bmatrix} \varphi_{w \to tractor} \\ \theta_{w \to tractor} \\ \psi_{w \to tractor} \end{bmatrix} \tag{3.1}$$

$$\text{Tractor rotation} = \begin{bmatrix} \varphi_{w \to tractor} \\ \theta_{w \to tractor} \\ 0 \end{bmatrix} \tag{3.2}$$

$$\text{Trailer rotation} = \begin{bmatrix} \varphi_{w \to tractor} + \varphi_{rel} \\ \theta_{w \to tractor} + \theta_{rel} \\ 0 + \psi_{rel} \end{bmatrix} \tag{3.3}$$

Where $\varphi, \theta, \psi_{w \to tractor,trailer}$ are measurements from IMUs on the respective vehicle frames.

3.1.1 Coordinate systems and naming convention

The sensors used for this project are described in Section 1.4. The vehicle configuration consists of 6 cameras and 2 IMUs. The IMUs generate data describing how each individual frame is rotated in relation to a world frame.

The hitch is defined along the same axis as the tractor's y-coordinate axis. Its purpose is to be a rigid offset between the tractor and trailer, inheriting its orientation, while also being the pivot point for the trailer. This approach was chosen to enable modeling of a wide variety of articulated vehicles, where their pivots may be positioned differently relative to the vehicle frames.

Each camera is defined with respect to either the tractor or trailer, where they inherit their parents orientation. This assumes that each camera stays in the same relative pose as when initially calibrating their extrinsics relative to the parent frame. The cameras are oriented according to the principles in Subsection 2.2.2.

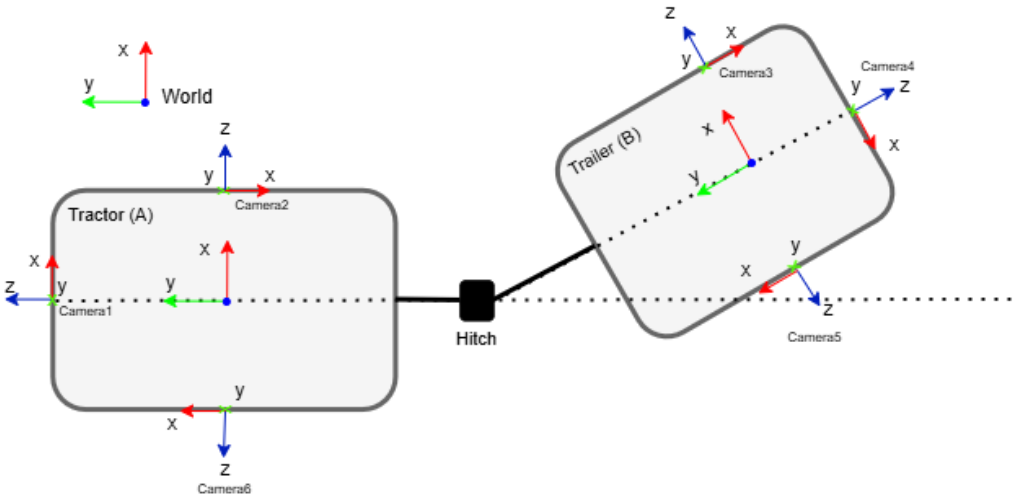


Figure 3.4: Camera placement and coordinate system of the tractor-trailer system during a positive articulation.

3.2 Simulation environment setup

The simulation environment was chosen after comparing two game engines, Unity and UnReal, where a virtual world could be built up as well as a representative vehicle. As no development of the actual project was to be carried out in the graphics engine itself, the choice was mainly based on usability.

Unity was selected based on the presumption of an easier setup for the authors. A simpler simulation environment was subsequently set up where the surroundings consisted of a racetrack that can be visualized in Figure 3.5.

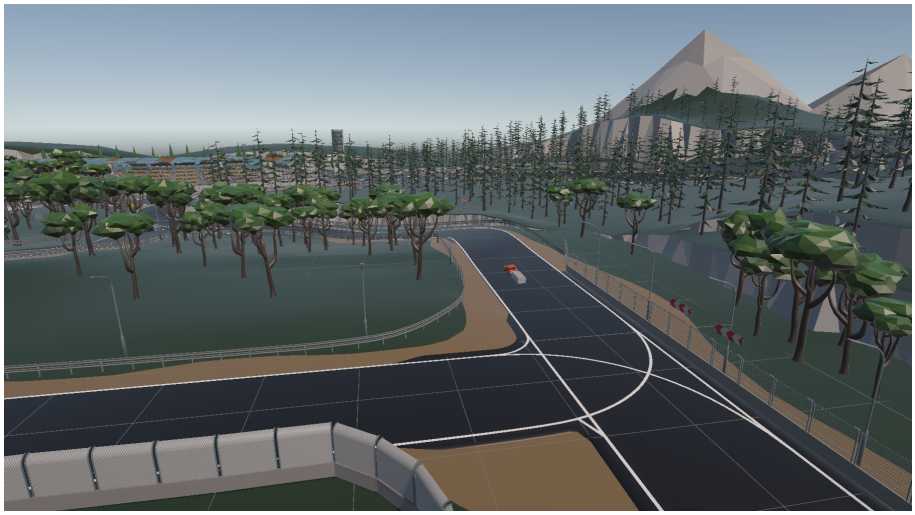


Figure 3.5: Image of the simulation environment used for initial development and testing.

The environment itself and the vehicle were downloaded from the Unity asset store.

The environment was designed by RCC Design [24] and the red car by Madtroll_studio [25].

Scripting was done within Unity to make sure that the movement of the vehicle was as representative of a real tractor-trailer system as possible.

3.2.1 Intrinsic and extrinsic calibration

The intrinsic matrix was obtained from the Unity environment by first knowing that it was set up as a pinhole camera model. Assuming this, the intrinsic parameters are fully defined by the camera's field of view and the image resolution, eliminating the need for explicit lens calibration or distortion modeling. The equation used to obtain this can be found in Equation 2.2. In Appendix A.1, it is possible to see the computed camera intrinsic matrix $\mathbf{K}$.

For the simulated setup, a field of view of $150° \times 150°$ and an image resolution of 480×480 pixels was used. This intrinsic matrix remained fixed throughout all experiments.

The camera extrinsics were obtained by building a script measuring their ground-truth rigid-body transforms from the vehicle frames in Unity, relative to the coordinate frame as seen in Figure 3.4. The extrinsics can be found in Appendix A.1.

3.2.2 Data acquisition

In Unity, cameras can be attached to objects and are thereby linked to their respective vehicle frame, as shown in Figure 1.2. Since no actual IMU game object exists in Unity, the world orientation of each vehicle frame's center of mass was extracted via a script to act as a virtual IMU.

The orientation data was transferred to Python after collection via a socket and a GStreamer[26] pipeline for the camera data.

3.2.2.1 Orientation noise

Since Unity's orientation data is not derived from a virtual sensor, the output is highly accurate and drift-free. This does not represent real-world sensors, so noise was added to the measurements. To best prepare for a real-world scenario and to obtain some kind of real representation, 2 CPAC IMUs (referred to as 1e and 1f) were tested for noise in their orientation output by lying still on a desk for 20 minutes. The output from this test is shown in Figure 3.6.

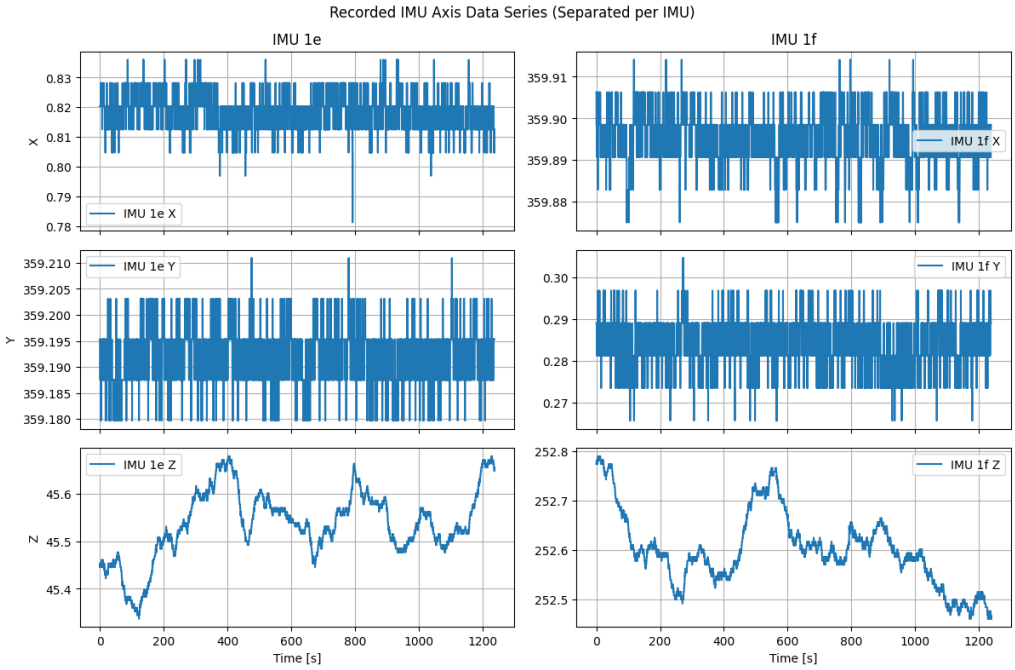


Figure 3.6: Orientation data from static CPAC IMUs.

It is worth noting that the orientation values as described in Section 2.6.1 have already run through an internal filter based on gyroscope and accelerometer data and are therefore in this output relatively low-noise. While extracting these plots, a script was run in Python, calculating the standard deviation of each axis for each of the IMUs, revealing the following results in Table 3.1.

Table 3.1: Result from noise testing of the CPAC IMU orientation data.

	std IMU 1e [deg]	**std IMU 1f** [deg]	**Average [deg]**
x-axis (φ)	0.005795	0.006096	0.0059455
y-axis (θ)	0.005445	0.005802	0.0056235
z-axis (ψ)	0.072143	0.073800	0.0729715

As seen by Table 3.1, the pitch and roll data display similar characteristics and the same noise level can thus be added to the simulated IMU measurement as a mean of these values.

The drift and residual noise around the z-axis (yaw) is however slightly more complicated as it contains a drift component. It was investigated and produced the following plot in Figure 3.7.

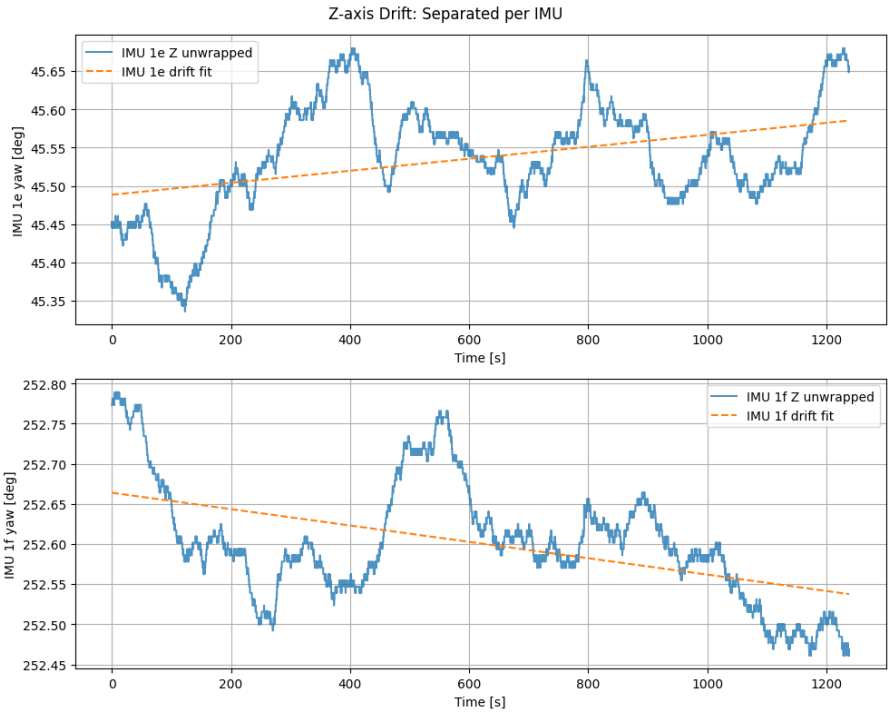


Figure 3.7: Plot showcasing the drift of the CPAC IMUs during a 20 min run.

Some data were obtained regarding the drift presented in Table 3.2. The residual standard deviation is the noise standard deviation after taking into account the effect of the drift on the noise. To find the drift, a line was fitted as visualized in Figure 3.7 with `numpy.polyfit()` in Python:

$$\hat{y}(t) = mt + b \tag{3.4}$$

where m and b are figures obtained by `numpy.polyfit()`. The residuals are calculated as the measurement subtracted from the fitted line

$$e_i = y_i - \hat{y}_i \tag{3.5}$$

And the residual standard deviation was then derived as

$$\text{residual std} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(e_i - \bar{e})^2} \tag{3.6}$$

Table 3.2: Yaw drift data from the IMU noise test run.

	IMU 1e	**IMU 1f**	**Average**
Global drift rate [abs(deg/min)]	0.0047	0.0061	0.0054
Residual std (after de-trend) [deg]	0.066533	0.064215	0.065374

The yaw axis exhibits a higher noise level than pitch and roll. It was thus set according to the mean of the residual standard deviations shown in Table 3.2.

As the unity data does not experience any drift, this behavior has to be added as well to the data received by the system. A yaw-drift model was configured in Python, which produces a very slowly varying drift to the signal to best imitate an actual IMU drift shown from the testing.

Lastly, the noise had to be rescaled as the IMU data is subtracted to form a difference in angle between the vehicle segments. As of the CPAC IMU tests, the standard deviation from each sensor is known and to be the same. Each sensor output is then defined as:

$$X_1 \sim \mathcal{N}(\mu_1, \sigma^2), \qquad X_2 \sim \mathcal{N}(\mu_2, \sigma^2) \tag{3.7}$$

and the difference between the sensor values is

$$D = X_1 - X_2 \tag{3.8}$$

and the variances, which are assumed to be independent are combined in Equation 3.8 as

$$\mathrm{Var}(D) = \mathrm{Var}(X_1) + \mathrm{Var}(X_2) = \sigma^2 + \sigma^2 = 2\sigma^2 \tag{3.9}$$

which means that the difference in standard deviation is:

$$\boxed{\sigma_D = \sqrt{2}\sigma} \tag{3.10}$$

Finally, the noise levels configured in the filter were set as follows:

Table 3.3: IMU expected noise.

Principal axes	**Added std** [deg]
φ	$\sqrt{2} \cdot 0.0057845 = 0.0081805$
θ	$\sqrt{2} \cdot 0.0057845 = 0.0081805$
ψ	$\sqrt{2} \cdot 0.065374 = 0.09245280$

3.2.3 Unity reference frames

Transferring the orientation introduced challenges, as the Unity environment uses a left-handed coordinate system, contrasting with the traditional right-handed standard.
In Unity, the standard is:

Table 3.4: Setup of unity reference frame.

x	right
y	up
z	forward

This meant that the raw data had to be converted before use in the main system for sensible calculations in roll, pitch and yaw, a left-hand rotation vector was recalculated to a right-hand as:

$$\begin{bmatrix} \varphi_{\text{real}} \\ \theta_{\text{real}} \\ \psi_{\text{real}} \end{bmatrix} = \begin{bmatrix} z_{\text{unity}} \\ -x_{\text{unity}} \\ -y_{\text{unity}} \end{bmatrix} \tag{3.11}$$

3.3 Real-world setup

To be able to test in a real-world scenario, there was initially a first step where the hardware was tested in small scale on a test rig at the office. This is described in Section 3.3.3, and when this worked as intended, the continuation was to evaluate on an even more realistic case with a full-scale wheel loader.

3.3.1 Differences from simulation

In the real environment, there are some limitations that were not present in the simulation. Primarily, the IMUs presented one of these differences in that upon start-up, they do not know their rotation in relation to each other in yaw, as the vehicle can be started at an arbitrary articulation angle. This would require calibration at startup or some other method, which is not within the scope of this master's thesis.

The cameras used in the real-world setup are different from the rendered images from Unity in a few important aspects. First of which is that the actual cameras described in Section 2.6.2 have a large amount of distortion, whereas the cameras in Unity are of a virtual kind and send out rectified images immediately. This prompts the use of intrinsic calibration described in Section 2.2.1.

Another notable difference is increased uncertainty in camera placement, as lens positions and orientations must be measured manually, introducing a significant error margin, particularly on larger vehicles. This can be solved with an extrinsic calibration, but it is also out of scope, it is an important note to consider.

An additional distinction is the resolution of the video frames, the IP-cameras delivers images in 1080p, whereas the Unity environment only delivers 480p images. This means that the chance of finding more features in the images increases, as well as the general visual quality of the final projection.

3.3.2 Intrinsic and extrinsic calibration

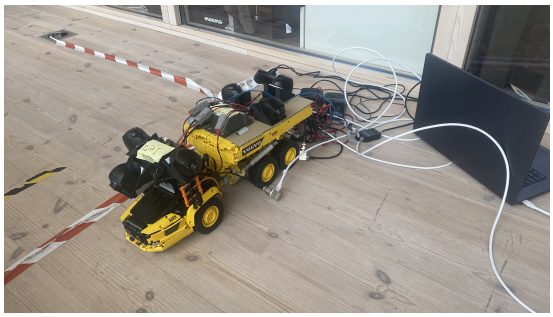
In the real-world setting, the intrinsics were calibrated with ChAruCo boards similarly to as described in Section 2.2.1. This gave the intrinsic matrix and camera distortion coefficients as can be seen in Appendix A.2.3.

The extrinsics were measured by hand on the vehicle, following the structure and convention as described in figure 3.4. The extrinsic measurements for the real-world setup can be found in Appendix A.2.

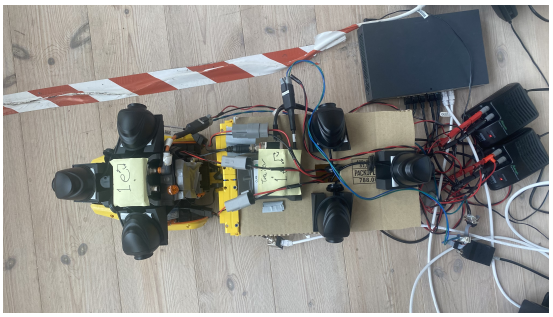
3.3.3 Small scale test rig

Before testing the system on a full-scale setup, a smaller version was developed with the same sensors for quick and easy testing and debugging as to bridge the

technical gap between simulation and real-world implementation. A test rig shown in Figure 3.8 was built on top of a LEGO TECHNIC construction vehicle, which has articulated movement in roll, pitch, and yaw.



(a) Isometric image of the test rig.



(b) Top down view of the test rig.

Figure 3.8: Test rig used for system development in the real-world case.

To be able to mount the cameras and IMUs in a stable manner, 3D printed holders and a plate were made and attached to the vehicle. The holders allowed the cameras to also be pointed in a negative angle towards the ground, which is important since the view aimed for is top down.

3.3.4 Realistic scale wheel loader

To support realistic system development, a real machine had to be used, which produces realistic vibrations, movement and camera placement. The testing was done on a Volvo L70G wheel loader where the cameras were placed evenly. Unfortunately, this machine is only articulated in yaw, which means that testing on shifting ground planes, requiring pitch and roll was not possible on this specific vehicle.

Figure 3.9: The wheel loader used for real-world data gathering and testing with mounted cameras and IMUs.

3.4 Orientation estimation and filtering

To be able to create a BEV interface of an articulated vehicle, the articulation angle between the vehicle frames is of high importance as it dictates how and where the camera frames are supposed to be stitched together. The IMUs already indirectly output this data, which is inherently filtered and exhibits low noise, as noted in Section 2.6.1. However, using yaw measurements to estimate trailer orientation introduces significant drift risk, which can skew the surround-view over time. The risk is due to yaw being unobservable from accelerometers and therefore accumulates drift, unlike roll and pitch which are gravity-referenced. To mitigate this, a Kalman filter, explained in Section 2.4.1, was set up using a complementary approach with visual orientation estimation from the cameras to help approximate the bias for yaw in the relative orientation vector (Equation 3.1). This combination is often referred to as a VINS system, briefly explained in Section 2.4.2. A general architecture for this was set up as presented in Figure 3.10.

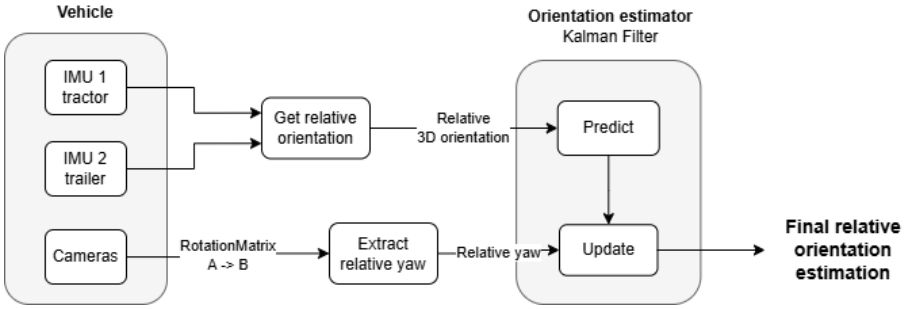


Figure 3.10: Set up of the orientation estimation algorithm.

The IMU data is combined with a lower frequency visual pose estimator that is run by utilizing the image frames from the side cameras, also referred to as camera index 2, 3, 5 and 6 in Figure 3.4.

3.4.1 IMU orientation estimation

The IMUs deliver an absolute three-dimensional angle measurement each that can be read over a CAN bus to a computer running the pose estimator. Pitch and roll are used directly from the tractor IMU, while the relative measurements in all dimensions are sent through the filter.

Obtaining initial measurements for pitch and roll, is straightforward as it is possible to always know these angles due to being derived from the consistent acceleration of gravity. However, at startup, the IMUs have no way of knowing how they are angled around the yaw-axis without calibration, as a magnetometer is not included in the measurements. This means that for this project, it is assumed that the vehicle is starting with 0 articulation in yaw, and the readings given by the IMU in yaw at startup are saved as an offset.

3.4.2 Visual orientation estimation

The visual pose estimator was built to utilize the overlap between the same side cameras on both sides of the vehicle to estimate the rotation in yaw between the vehicle frames. The advantage of this approach is that there is always at least one side with a considerable amount of overlap, when the other diminishes. This means that there is always potential in searching for features, giving a more stable estimation of the angle between the vehicle frames. The overall functionality can be described as shown in Figure 3.11:

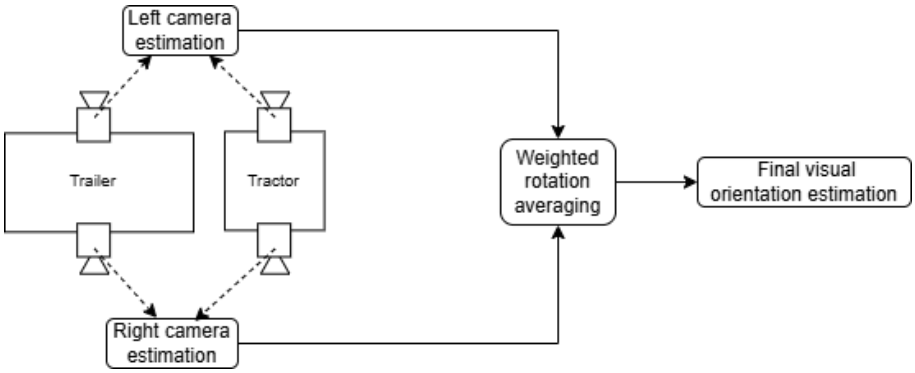


Figure 3.11: High-level overview of how the visual estimator works.

Each of the two camera pairs create their own measurement in the form of a rotation matrix $\mathbf{R}_A^B$ of how the vehicle frames are rotated in all three dimensions. These two measurements are then refined through weighted rotation averaging, given their inlier-ratio from RANSAC, creating an optimized measurement sent through to the rest of the system.

The algorithm for extracting the rotations of one camera in relation to the other is based on functions within the OpenCV library and is visualized in Figure 3.12. This includes relevant functions such as reading images, extracting image data, and performing image-related calculations based on the said image data.

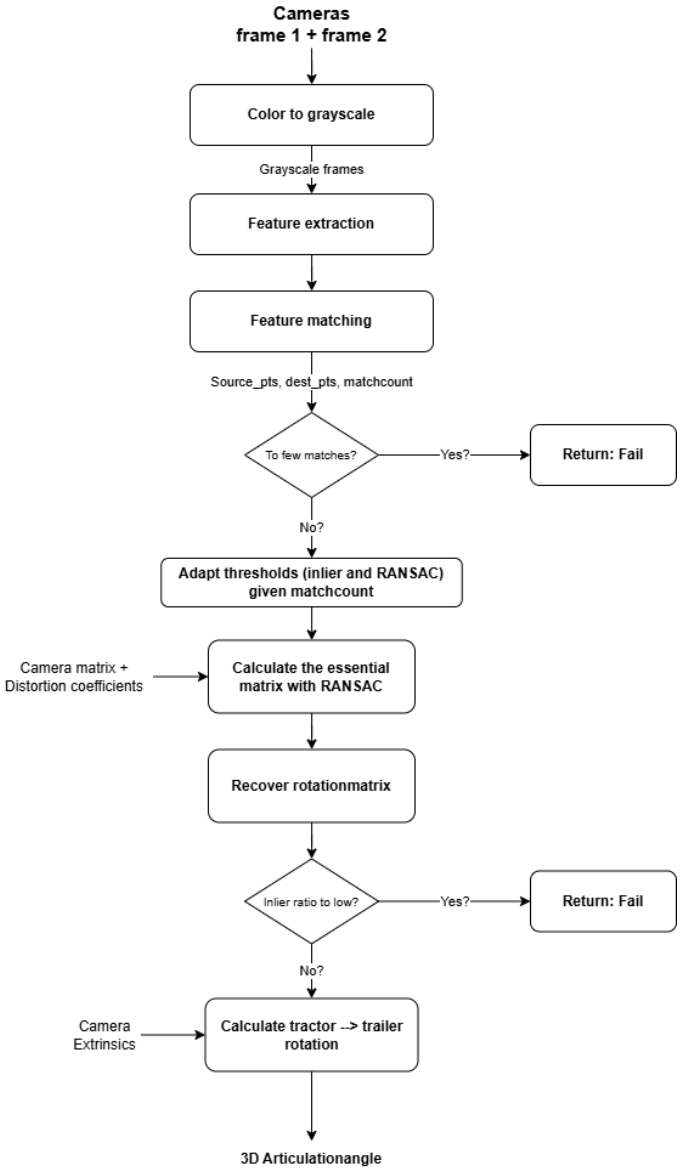


Figure 3.12: Flowchart of the visual orientation estimation algorithm on a single side of the vehicle.

The features of the images are first located using a feature extractor through OpenCV, more specifically, the Oriented FAST and Rotated BRIEF (ORB) for the real-world scenario, and Scale-Invariant Feature Transform (SIFT) for simulation. These methods return keypoints alongside their respective descriptors. A brute force matcher in OpenCV is then used to identify corresponding keypoints between images based on a cost function. The matcher returns a ranked set of matched features, sorted by strength.

At this point, it is possible to check the total number of matches, which is a good indicator of whether the images received from the cameras are similar enough to be used at all. A simple check is done here with an exit condition in the algorithm if the match count is too low.

After the matches have been found, triangulation can be performed using RANSAC in order to find the relative position between the viewpoints. This can be done by using an essential matrix, which is a description of how points in one image relative to another relate to each other in terms of rotation and translation.

$$(\mathbf{y}')\,\mathbf{E}\,\mathbf{y} = 0 \tag{3.12}$$

Finding the essential matrix is in this project done by using the OpenCV functions `cv.findEssentialMat()` and `cv.recoverPose()`, both described in Sections 2.3.1 and 2.3.2, which find the rotation between the camera angles using RANSAC and the camera's intrinsics described in Section 2.2.1.

RANSAC is an algorithm that, in this use case, takes the matches between the frames and fits a model corresponding to the points. The subset of matches given to RANSAC that supports the returned model is called inliers and is used as an indication of how certain one can be of the transformation returned. An exit condition is implemented here as well, which is based on the ratio of inliers supporting the transformation found by RANSAC.

For the algorithm to be able to give the actual vehicle frame relations and not only the individual cameras' relation to each other, calculations of transformations were performed so that the correct measurement is obtained as in Equation 3.13. The frame and camera indices are given the convention in Figure 3.4.

$$\begin{aligned} &\text{Right side:} \\ &\quad R_A^B = R_A^{c2}\, {R_{c2}^{c3}}^{\top}\, {R_B^{c3}}^{\top} \\ &\text{Left side:} \\ &\quad R_A^B = R_A^{c6}\, {R_{c6}^{c5}}^{\top}\, {R_B^{c5}}^{\top} \end{aligned} \tag{3.13}$$

The rotation matrices to the cameras with respect to the frames A and B are known beforehand, given extrinsic calibration.

3.4.2.1 Weighted rotation averaging

To combine the two visual rotation estimates, a weighted rotation averaging approach is used. Each camera pair produces an estimate of the relative rotation between the vehicle frames, denoted $\mathbf{R}_i$, together with a confidence weight w_i derived from the RANSAC inlier ratio.

A simple weighted average in axis-angle space provides an initial estimate. However, due to the nonlinear nature of rotations and frequent inconsistencies between the observations, the estimate is further refined iteratively.

The approach can be interpreted as a simplified form of rotation averaging, where a single rotation estimate is obtained from multiple observations by minimizing their

disagreement on SO(3), following the formulation described by Hartley et al. [27]. The residual rotation is computed in axis-angle form, and the estimate is refined iteratively.

At each iteration, the rotational difference between the current estimate $\mathbf{R}$ and each observation $\mathbf{R}_i$ is computed as

$$\mathbf{R}_{\text{err},i} = \mathbf{R}_i \, \mathbf{R}^\top \tag{3.14}$$

which represents the rotation required to align the estimate with the observation. This error is converted into an axis-angle vector using the logarithm map, and a weighted average correction is computed as

$$\Delta = \frac{\sum_i w_i \, \text{Log}(\mathbf{R}_i \mathbf{R}^T)}{\sum_i w_i} \tag{3.15}$$

The estimate is then updated by applying this correction, and the process is repeated until Δ is small enough. To improve robustness, a Huber-type weighting reduces the influence of large residuals, and a temporal prior from the previous timestep is included to ensure smooth frame-to-frame behavior.

3.4.3 Filtering and sensor fusion

A regular Kalman filter is used to combine the data, filter IMU noise, and incorporate correction terms from the visual orientation estimation. The state vector was set up as in Equation 3.16, where the states φ_{rel}, θ_{rel}, ψ_{rel} and $b_{\psi_{rel}}$ are expressed in degrees and $\dot{\varphi}_{rel}$, $\dot{\theta}_{rel}$ and $\dot{\psi}_{rel}$ representing their derivatives with units in degrees/s.

The relative angle states represents the unified IMU data from both sensors and provide a single way of representing the rotated state of the hitch. With this setup, it is easier to model the movement while also keeping the number of states down.

$$\mathbf{x} = \begin{bmatrix} \varphi_{rel} \\ \theta_{rel} \\ \psi_{rel} \\ \dot{\varphi}_{rel} \\ \dot{\theta}_{rel} \\ \dot{\psi}_{rel} \\ b_{\psi_{rel}} \end{bmatrix} \tag{3.16}$$

The information from the change rate derivatives helps to guide the filter through the process model of which direction it is reasonable that the joint is moving. Additionally, a bias state is added to the vector, which is used to store the perceived offset in yaw that is between the IMU measurements and the true state. The motivation for this state is the unavoidable drift in yaw that comes when running an IMU for a longer period of time.

Due to the physical constraints of the sensor, it is in the yaw dimension where the IMU might suffer. On the other hand, it is very accurate in pitch and roll, therefore

the need for correction in the other two dimensions is not necessary to the same extent.

3.4.3.1 Motion and observation model

In this filter, the IMU data is sent through the prediction step and thereafter the update step together with the most recent measurement values. If there is also a new value from the visual orientation estimation algorithm, an additional update is executed with a slightly different observation matrix that does not inject the bias state, as the visual data does not experience drift.

The motion model shown in Equation 3.19 incorporates knowledge about how the hitch is suspected to move and is used in the prediction step. It is for example not reasonable to have an articulation in yaw of $+50°$ and half a second later $-50°$. The equations in the model for the angular states follow a general form of motion as the movement within the hitch is considered relatively simple:

$$\alpha_{t+1} = \alpha_t + \Delta t\, \dot{\alpha}_t \tag{3.17}$$

The states for roll-, pitch-, and yaw rate follow a spring model so that they can be realistically regulated in the filter through stiffness and damping.

$$\begin{aligned} &\text{Roll and pitch:} \\ &\dot{\alpha}_{t+1} = (1 - d_{\varphi,\theta})\dot{\alpha}_t - s_{\varphi,\theta}\, \alpha_t \\ &\\ &\text{Yaw:} \\ &\dot{\alpha}_{t+1} = (1 - d_{\psi})\dot{\alpha}_t \end{aligned} \tag{3.18}$$

The α represents the angle and $\dot{\alpha}$ angle rate, d are for damping and s the stiffness terms that can be tuned to tell the filter what is a realistic movement of the hitch. The motion model contains damping and stiffness terms which help tune the model to more accurately represent the movement in an actual hitch. They are set as parameters in damping/stiffness per second and then multiplied by the length of the timestep before altering the angle rates.

$$F_k = \begin{bmatrix} 1 & 0 & 0 & \Delta t_s & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t_s & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t_s & 0 \\ -s_{\varphi,\theta} & 0 & 0 & 1 - d_{\varphi,\theta} & 0 & 0 & 0 \\ 0 & -s_{\varphi,\theta} & 0 & 0 & 1 - d_{\varphi,\theta} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 - d_{\psi} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.19}$$

The observation model used for updating the angular states with IMU measurements is shown in Equation 3.20, where the bias state is introduced to counter the possible drift in yaw from the IMU.

$$H_{k,IMU} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.20}$$

The secondary observation model used for the visual updates is essentially the same as the previous, but without the bias, as shown in Equation 3.21.

$$H_{k,vis} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.21}$$

3.4.3.2 Kalman equations

For the Kalman equations in the case where there is no visual orientation update, these equations are performed in one Kalman step. The first prediction step only performs a state and covariance update based on the previous state and the motion model.

$$\hat{\mathbf{x}}_{k|k-1} = F_k \, x_{k-1|k-1} \tag{3.22}$$

$$P_{k|k-1} = F_k \, P_{k-1|k-1} \, F_k^\top + Q_k \tag{3.23}$$

After a predicted step has been updated, the innovation is calculated, which tells the filter what a reasonable evolution of the state is and how much the most recent measurement $\mathbf{z}_k$ differs from it. The innovation covariance S_k is also calculated for later use.

$$\tilde{\mathbf{y}}_k = \mathbf{z}_k - H_k \, \hat{\mathbf{x}}_{k|k-1} \tag{3.24}$$

$$S_k = H_k \, P_{k|k-1} H_k^\top + R_k \tag{3.25}$$

The Kalman gain is calculated next, which will allow the filter to update the previous state estimation as well as possible, given the measurements. Lastly, the covariance is updated using the Joseph form, a numerically stable variant of the standard update [21].

$$K_k = P_{k|k-1} H_k^\top S_k^{-1} \tag{3.26}$$

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + K_k \, \tilde{y}_k \tag{3.27}$$

$$P_{k|k} = (I - K_k H_k) P_{k|k-1} (I - K_k H_k)^\top + K_k R_k K_k^\top \tag{3.28}$$

After that, the filter has finalized estimating the states, a refined and drift-proof version of the relative orientation vector is obtained and can be used to get the trailer's orientation i.e. its absolute world orientation as described in the Vehicle model section in Equations 3.2 and 3.3. Thus, the orientations used further are as follows in Equations 3.29 and 3.30.

$$\text{Tractor rotation} = \begin{bmatrix} \varphi_{w \to tractor} \\ \theta_{w \to tractor} \\ 0 \end{bmatrix} \tag{3.29}$$

$$\text{Trailer rotation} = \begin{bmatrix} \varphi_{w \to tractor} + \varphi_{rel,filter} \\ \theta_{w \to tractor} + \theta_{rel,filter} \\ 0 + \psi_{rel,filter} \end{bmatrix} \tag{3.30}$$

The filter subscript variables are the filtered roll, pitch and yaw and the "$w \to tractor$ subscript variables are the immediate and absolute IMU values from the tractor frame.

3.4.4 Conversion from Euler angles to rotation matrix

The creation of the bird's-eye-view requires the Euler angles described as a rotation matrix. The estimated roll, pitch and yaw angles obtained from sensor data are therefore converted into a rotation matrix using the ZYX Euler angle conversion defined in Section 2.5.

For each vehicle frame, the corresponding rotation matrix from the world frame to the local plane frame is computed as:

$$\mathbf{R}_{\mathrm{w}}^{\mathrm{p}} = \mathbf{R}_z(\psi)\, \mathbf{R}_y(\theta)\, \mathbf{R}_x(\varphi) \tag{3.31}$$

where ψ, θ and φ are the elements from the absolute orientation vectors calculated in Equations 3.29 and 3.30. This rotation matrix will be used to define the orientation of the local ground plane, enabling homography computation and image warping in the bird's-eye-view projection pipeline.

3.5 Creation of the bird's-eye-view

This section describes the full bird's-eye-view generation pipeline, from raw camera images to a stitched BEV projection of the environment and a representation of the articulated vehicle. The pipeline is visualized in Figure 3.13 and consists of image undistortion, homography computation onto locally defined ground planes, projection onto a metric BEV grid, adaptive seam alignment and final image blending. The tractor and trailer frames use their individually computed rotation matrices from Section 3.4.4 to derive respective ground planes that adapt to vehicle articulation and uneven terrain.

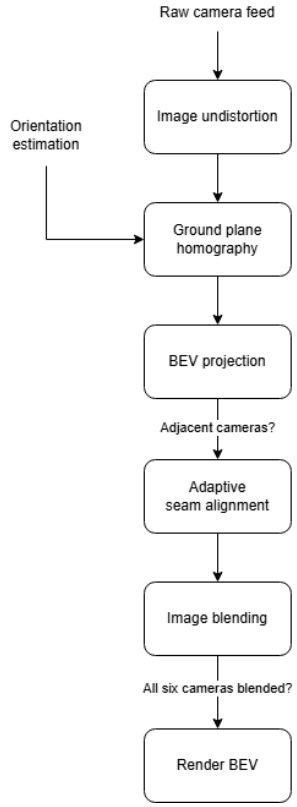


Figure 3.13: BEV creation flowchart.

3.5.1 Undistortion of camera images

Raw images that are acquired from wide-angle and fisheye cameras need to use the distortion coefficients as from Section 2.2.1 in order to remove the radial and tangential distortions. This is needed as the inverse perspective mapping, homography projections, and stitching all assume geometrically correct images.

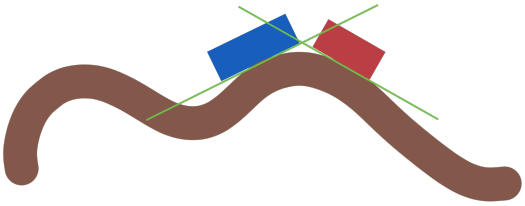
For each camera and image resolution, undistortion is implemented as a two-step process. First, a pair of rectification maps is generated using OpenCV's remapping function. The function `cv2.fisheye.initUndistortRectifyMap` is used, which supports the equidistant fisheye distortion model and is described in Section 2.3.4. These rectification maps are only computed once per camera and image resolution and then cached. In the second step, the actual undistortion is performed using `cv2.remap`, which applies the precomputed rectification maps to the input image. The result is an undistorted image that preserves straight-line geometry.

3.5.2 Tilting ground plane homography calculation

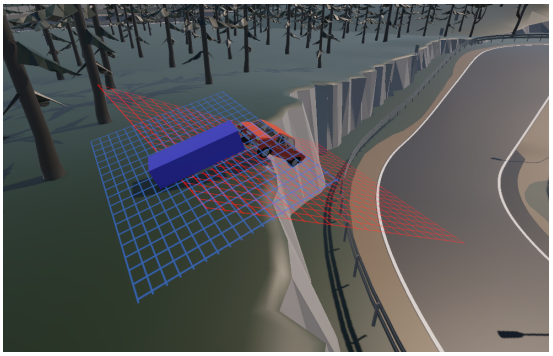
The purpose of the rotation computations in this section is to express each camera pose relative to a locally tilted ground plane, rather than a globally horizontal ground plane based on the previous orientation estimation. This enables IPM to continue to be geometrically valid when the vehicle experiences roll and pitch due

to varying terrain.

Separate orientation data from the two vehicle frames was used, as illustrated in Figure 3.14, where the tractor and trailer each maintain independent local plane representations. Due to articulation and uneven terrain, each frame may experience independent roll and pitch, requiring definition from its own local ground plane. The planar homography theory in Section 2.2.3 assumes world points lie on a $Z = 0$ plane; this assumption only holds when the observed surface is strictly horizontal relative to the camera. The pose of a plane in the world frame can be parameterized by a rotation matrix $\mathbf{R}_w^p$, as defined in Equation 3.31, and an origin $\mathbf{o}_p$ that specifies the location of the plane's local coordinate frame.



(a) Simple illustration showing the concept of changing ground planes from the side. The green lines illustrate each of the vehicle frames independent planes.



(b) Ground plane overlay of the vehicle frames in the simulation environment.

Figure 3.14: Independent tilting ground planes for the towing vehicle and trailer.

To now account for these variations, we define each ground plane in terms of its own coordinate system $\mathbf{R}_{\mathrm{w}}^{\mathrm{p}}$ and a plane origin $\mathbf{o}_p$. This is used to describe the plane's orientation and height relative to a set world frame, which is defined from the tractor frame's coordinate system. By formulating the homography with this approach, the BEV transformation aims to become more robust to local tilts. The following method is therefore implemented to transform the camera pose from the world frame into their chosen ground planes.

By transforming the camera's rotation matrices dynamically from its initial extrinsically calibrated camera to world matrix $\mathbf{R}_{\mathrm{c}}^{\mathrm{w}}$, into the camera to plane rotation, the following is done as in equation 3.32. This expresses the cameras' rotation from the local ground plane rather than the world frame.

$$\mathbf{R}_{\mathrm{c}}^{\mathrm{p}} = \mathbf{R}_{\mathrm{w}}^{\mathrm{p}}\,\mathbf{R}_{\mathrm{c}}^{\mathrm{w}} \tag{3.32}$$

The camera to plane translation vector is defined as $\mathbf{t}_{\mathrm{c}}^{\mathrm{w}}$, and the plane origin as $\mathbf{o}_p$. The equation describes the displacement vector from the plane origin to the specified camera, but is still expressed in the world coordinate system. To express

this displacement in the plane frame, it is transformed using the world-to-plane rotation matrix $\mathbf{R}_{\mathrm{w}}^{\mathrm{p}}$. The mathematical expression is shown in equation 3.33.

$$\mathbf{t}_c^p = \mathbf{R}_{\mathrm{w}}^{\mathrm{p}} \left(\mathbf{t}_{\mathrm{c}}^{\mathrm{w}} - \mathbf{o}_p \right) \tag{3.33}$$

In order to construct the planar homography, the rotation matrix needs now to be mapped from the plane to the camera as per the theory from equation 2.6. However, the previous derivation expressed the camera pose relative to the plane as $\mathbf{R}_{\mathrm{c}}^{\mathrm{p}}$ and $\mathbf{t}_{\mathrm{c}}^{\mathrm{p}}$.

To get the opposite mapping, a rotation matrix can be transposed as it is an orthogonal matrix. The camera to plane orientation matrix is therefore transposed as in Equation 3.34.

$$\mathbf{R}_{\mathrm{p}}^{\mathrm{c}} = (\mathbf{R}_{\mathrm{c}}^{\mathrm{p}})^{\top} \tag{3.34}$$

The same process is performed for the translation vector, where it is being transformed from the camera to the plane. The vector needs to point from the plane origin to the camera origin, hence the negation.

$$\mathbf{t}_{\mathrm{p}}^{\mathrm{c}} = -\mathbf{R}_{\mathrm{p}}^{\mathrm{c}} \mathbf{t}_{\mathrm{c}}^{\mathrm{p}} \tag{3.35}$$

The homography from plane to image is a mapping using the first and second columns of the rotation matrix $\mathbf{R}_{\mathrm{p}}^{\mathrm{c}}$. We only take the first two columns of $\mathbf{R}_{\mathrm{p}}^{\mathrm{c}}$ because points on the plane have no degree of freedom along the plane normal. The third column corresponds to motion out of the plane, since all planar points have zero coordinate in this direction, its contribution vanishes. To account for the projection related to the camera intrinsics, the camera intrinsics matrix $\mathbf{K}$ is applied to the homography, incorporating both the camera focal lengths and principal point to map the coordinates from the camera plane into the image plane.

$$H_{\mathrm{plane}}^{\mathrm{img}} = \mathbf{K} \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t}_{\mathrm{p}}^{\mathrm{c}} \end{bmatrix} \tag{3.36}$$

To then get homography mapping from the image to the plane, the homography is inverted with an inverse perspective mapping in Equation 3.37.

$$H_{\mathrm{img}}^{\mathrm{plane}} = \left(H_{\mathrm{plane}}^{\mathrm{img}} \right)^{-1} \tag{3.37}$$

3.5.3 BEV grid definition

The bird's-eye-view representation is defined on a fixed two-dimensional grid, which is aligned with the ground plane. It is implemented with a constant metric resolution, which allows for a consistent interpretation of the environment. The matrix $\mathbf{T}_{\mathrm{world}}^{\mathrm{pix}}$ is a transformation from world-plane coordinates into the bird's-eye-view pixel coordinates. The negative sign in the vertical scaling term accounts for the difference between the world-plane coordinate system and the image coordinate convention, the pixel origin is located in the upper-left corner and the axis increases downward.

$$\mathbf{T}_{\text{world}}^{\text{pix}} = \begin{bmatrix} \frac{1}{m_{\text{px}}} & 0 & -\frac{x_{\min}}{m_{\text{px}}} \\ 0 & -\frac{1}{m_{\text{px}}} & \frac{y_{\max}}{m_{\text{px}}} \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.38}$$

In addition, w and h represent the width and height of the BEV.

$$w = \left\lceil \left(x_{\max} - x_{\min}\right) \frac{1}{m_{\text{px}}} \right\rceil, \qquad h = \left\lceil \left(y_{\max} - y_{\min}\right) \frac{1}{m_{\text{px}}} \right\rceil, \tag{3.39}$$

3.5.4 Image warping to tilting ground planes

To finalize the transformation chain, a homography mapping from the local plane coordinate system is required. Under the planar constraint that $Z_p = 0$, the resulting plane-to-world mapping reduces to a two-dimensional homogeneous transformation composed of the first two columns of $\mathbf{R}_{\text{p}}^{\text{w}}$ and the planar components of the plane origin. This aligns with the planar homography formulation described in Hartley et al. (Chapter 13) [19].

$$\mathbf{R}_{\text{p}}^{\text{w}} = \begin{bmatrix} r_{11}^{pw} & r_{12}^{pw} & r_{13}^{pw} \\ r_{21}^{pw} & r_{22}^{pw} & r_{23}^{pw} \\ r_{31}^{pw} & r_{32}^{pw} & r_{33}^{pw} \end{bmatrix} = (\mathbf{R}_{\text{w}}^{\text{p}})^{\top} \tag{3.40}$$

The vector

$$\mathbf{o}_p = [o_{p,x},\, o_{p,y},\, o_{p,z}]^{\top}$$

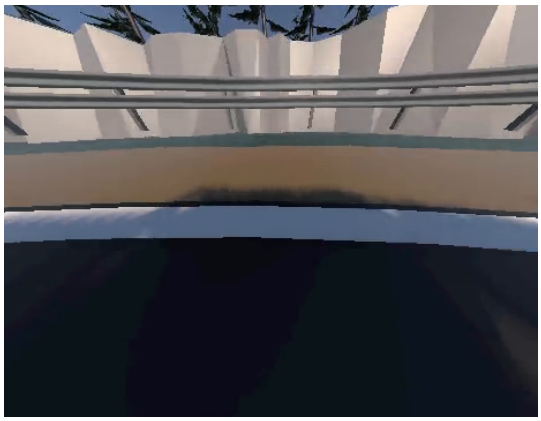
denotes the origin of the plane coordinate system expressed in the world frame. Since the homography operates in the plane $Z_p = 0$, only the planar components $o_{p,x}$ and $o_{p,y}$ are included in the homogeneous transformation.

$$H_{\text{plane}}^{\text{world}} = \begin{bmatrix} r_{11}^{pw} & r_{12}^{pw} & o_{p,x} \\ r_{21}^{pw} & r_{22}^{pw} & o_{p,y} \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.41}$$

To obtain the finalized homography mapping from image pixels through the tilted ground plane, into world coordinates, and finally into the discretized BEV pixel grid, the previous methods from subsections 3.5.2 and 3.5.3 are used as in Equation 3.42.

$$H_{\text{img}}^{\text{bev}} = T_{\text{world}}^{\text{pix}}\, H_{\text{plane}}^{\text{world}}\, H_{\text{img}}^{\text{plane}}. \tag{3.42}$$

Now it is possible to correctly warp each image by applying the individually computed $H_{\text{img}}^{\text{bev}}$ to each camera image, this was done using `cv.warpPerspective()` as described in Section 2.3.3. In Figure 3.15, one camera image feed from the simulated environment is shown and another showing the BEV projection.



(a) Image feed from simulation environment.

(b) Projected image feed from the simulation environment in the BEV.

Figure 3.15: BEV projection from image.

Thereafter, each image has been properly projected onto the ground plane with each camera's intrinsics and extrinsics.

3.5.5 Adaptive seam alignment

Since the vehicle frame geometry changes with the articulation angle, affecting each camera's field of view, the stitch line must dynamically adjust to preserve valuable information, as the relative geometry between cameras changes continuously. A fixed seam placement is therefore not sufficient, as it may intersect regions containing important scene information. Following Wei et al. [16], the seam is computed using the articulation angle as a basis. The author's algorithm calculates the seam line as the average of the adjacent cameras' current articulation angles relative to the world coordinate frame, it uses the geometric configuration of the camera system rather than image-based data, making it suitable for stability and computational efficiency. This approach is also based on the observation that objects from the nearest camera are typically represented with higher spatial resolution, as they occupy more pixels in the image.

Let the world positions of the two adjacent cameras be $\mathbf{p}_i$ and $\mathbf{p}_j$. Their midpoint is

$$\mathbf{p}_{\text{mid}} = \frac{\mathbf{p}_i + \mathbf{p}_j}{2}. \tag{3.43}$$

Each camera has an associated rigid-body transform $\mathbf{T}_i \in SE(3)$, whose rotation component $\mathbf{R}_i \in SO(3)$ defines the camera's forward (Z-axis) direction in world coordinates:

$$\mathbf{f}_i = \mathbf{R}_i \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}. \tag{3.44}$$

These forward directions were projected onto the ground plane and their average was computed:

$$\bar{\mathbf{f}} = \frac{\mathbf{f}_i^{(xy)} + \mathbf{f}_j^{(xy)}}{2}, \tag{3.45}$$

where $\mathbf{f}_i^{(xy)}$ denotes the first two components (x,y) of $\mathbf{f}_i$.

Finally, the averaged direction was normalized to obtain the dynamic seam direction:

$$\hat{\mathbf{f}} = \frac{\bar{\mathbf{f}}}{\|\bar{\mathbf{f}}\|}. \tag{3.46}$$

The seam line between cameras i and j is then defined by the point $\mathbf{p}_{\text{mid}}$ and the direction $\hat{\mathbf{f}}$, adapting continuously with the current articulation angle, it defines a straight seam line in the BEV plane around which blending weights are applied. How this can ultimately look is represented in Figure 3.16. The figures are using data from the real-world tractor-trailer setup.

(a) Seams with no articulation. **(b)** Seams in a turning scenario.

Figure 3.16: Representation of the adaptive seams in a real scenario where the white fields are blend zones centered on the seam line.

3.5.6 Image blending

The projected images are blended using a distance-based feathering approach, inspired by the method described by Szeliski [28], it was implemented by using `cv.distanceTransform()` as described in Section 2.3.5. Distance-based feather blending was chosen due to its low computational complexity and robustness in the presence of geometric misalignment. More advanced blending techniques, such as multiband or gradient-domain blending, can reduce intensity discontinuities but are

computationally more expensive and may introduce excessive smoothing, which is undesirable for a real-time, operator-facing visualization. In Szeliski's formulation, the contribution of each image is weighted according to a blending weight, but his approach relies on a center-focused weighting scheme. In contrast, this method uses distance-based weighting from the edges of each image mask, which ensures smoother transitions in only overlapping regions and reduces visible seams. This reduces unnecessary blurring in non-overlapping areas.

The blending is computed using the following equation:

$$C(x) = \frac{\sum_k w_k(x)\, \tilde{I}_k(x)}{\sum_k w_k(x)} \tag{3.47}$$

where $C(x)$ is the final blended color at pixel position x, $\tilde{I}_k(x)$ is the color contribution from image k, and $w_k(x)$ is the spatially varying weight that decreases as the pixel approaches the boundary of image k.

3.5.7 BEV rendering viewer

The bird's-eye view is defined on a fixed planar grid where all camera images are projected. This gives us a two-dimensional metric coordinate system aligned with the vehicle's ground plane. In Figure 3.17, a full representation of a frame from the renderer can be seen.

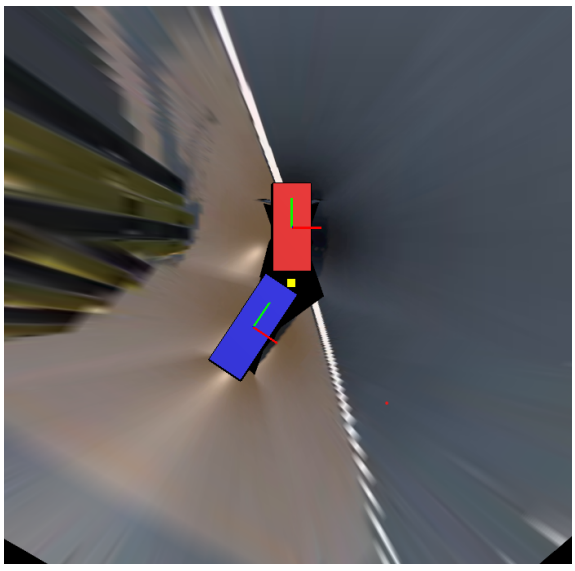


Figure 3.17: A frame shown from the BEV rendering viewer using simulated data.

In the implemented viewer, the BEV ground plane is rendered with fixed bounds for the projection, which is possible to customize and is often dependent on the size of the vehicle. This provides a basis for all the previous operations, including warping, seam adjustment and image blending from Sections 3.5.4, 3.5.5 and 3.5.6, and a draft visualization of the articulated vehicle model in relation to its environment. This

articulated vehicle model is rendered as two vehicle frames; the tractor is rendered as a red cuboid, and the trailer as a blue cuboid.

3.6 Evaluation

The evaluation was set up using simulated Unity data alongside a real-world scenario. This combination was chosen to allow for controlled testing of specific system properties. Simulated tests were conducted to generate sloped terrain data, which could not be replicated in physical tests due to the test vehicle's inability to tilt its frames independently in pitch and roll. Another aspect that is difficult to evaluate in the real world is the filter's drift mitigation. Unity allowed for arbitrary drift injection, simplifying the assessment of this property. Since the project scope excludes precise intrinsic and extrinsic calibration (Section 1.4), BEV performance will not be evaluated analytically, as results would be heavily influenced by calibration errors. Instead, the results will be based on BEV images, where it will be possible to evaluate them based on the visual information they provide.

To ensure a systematic evaluation despite the lack of ground truth data, the presented scenarios were selected to represent different operating scenarios relevant for articulated vehicles, including flat terrain, uneven terrain, articulation between the vehicle frames and orientation drift from the IMUs about the yaw axis.

3.6.1 Simulated environment

In the simulated environment, it was set up as to use true orientation data given from the Unity engine. This was done in order to isolate the effect of the tilting ground plane and not be affected by external noise varieties.

The tests were done in three different settings, one showing a planar area similar to a parking lot, a sloped road with road markings, and a few other adjacent objects. The other shows an uneven forest terrain with trees.

For BEV generation, two homography-based approaches were evaluated. The first is a conventional flat-ground-plane approach, where the scene is projected onto a single horizontal planar homography, which is derived from the calibrated camera intrinsics and extrinsics, following the same standard planar homography formulation described by Hartley and Zisserman [19]. The second approach uses the developed method in Section 3.5.2. The performance of the tilted-plane approach is evaluated relative to the flat-plane baseline in both simulation scenarios and to a rendered BEV as seen from above the vehicle in Unity from the same moment.

3.6.2 Real-world setup

The rest of the system was ultimately tested in a real-world environment for more realistic and representative results. The raw data from the IMUs were extracted

as well as the camera streams from the mounted IP-cameras, while the vehicle performed several different movements.

- Standing still, without articulation.
- Driving forward, without articulation.
- Standing still and articulating in both directions around yaw.
- Driving in a figure-8.
- Reversing next to another vehicle (parking scenario).

From these scenarios, the entire system can be evaluated as well as the subsystems individually.

For the real-world setup, computational performance was also evaluated in order to see how appropriate the final system was for running in real-time on less powerful hardware. This was performed by checking the frame rate while running on both recorded data and in a live scenario. An average fps of each run was calculated and multiple runs were executed to get an average of the specific setup, recorded data and live-streamed.

The testing was done on two different machines, running the same scenarios:

1. **Lenovo Thinkpad**, Processor: Intel Core i9 13900H, GPU: Nvidia RTX 2000 Ada
2. **Dell Pro Max 18**, Processor: Intel Core Ultra 7 265 HX, GPU: Nvidia RTX PRO 3000 Blackwell

The view rendered in all scenarios is a 512×512 window with the view constituting 360×480.

4

Results

The following results section is split up into the final results obtained in each of the two environments: simulation and real-world.

4.1 Simulated environment

The simulated environment differed from the real-world setup in several respects, particularly with regard to texture richness and camera resolution. However, simulation facilitated testing of specific subsystems, particularly the handling of tilting ground planes and the drift compensation capabilities of the VINS Kalman filter.

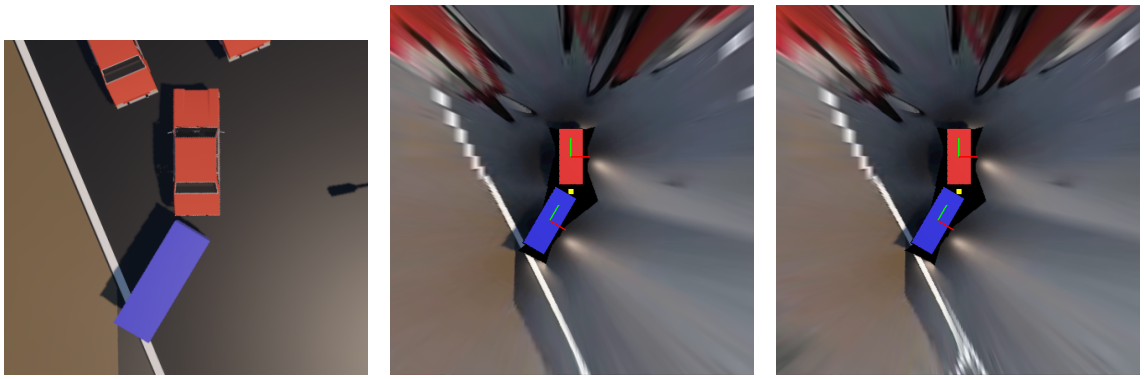
4.1.1 Countering IMU-drift

When additional drift was added to the sensor, the designed filter compensated for the offset with the help of the camera updates via the drift bias, as shown in Figure 4.1. The IMU data was driving the estimations together with corrections from the camera-based orientation estimation.

Figure 4.1: Test in Unity with added drift and noise for demonstrating the drift countering capabilities of the filter.

4.1.2 Effect of tilting ground planes

In the subfigures in 4.2, there are three figures showing the tractor-trailer vehicle setup on a flat road with two adjacent cars. The table in 4.1 shows that both vehicle frames have minimal tilt in both roll and pitch. Here, the trailer frame is rotated to the left relative to the tractor.



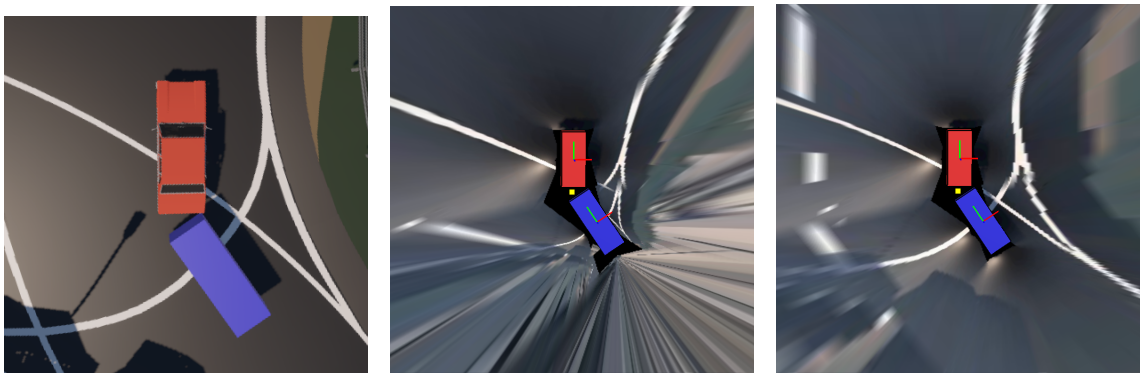
(a) True BEV. **(b)** Flat-plane BEV. **(c)** Tilted plane BEV.

Figure 4.2: Comparison of BEV projections using a flat-plane homography, the proposed tilted-plane homography, and the reference view while traversing a tilted road.

Table 4.1: Ground-truth orientation angles used in the simulated environment related to Figure 4.2.

Frame	Roll [deg]	Pitch [deg]	Yaw [deg]
Tractor frame	-0.001227	0.001932	0.0000
Trailer frame	-1.267520	0.013200	-30.264430

The three subfigures in 4.3 show how the true bird's-eye-view from the Unity engine looks compared to when using the flat-plane approach and the tilted plane approach, while traversing up a tilted road with visible road markings. The flat-plane projection introduces visible distortion inversions, which can be linked to the increasing pitch and roll angles listed in Table 4.2. The tractor frame has a yaw of 0 degrees, as it shares the same z-axis as the world frame for consistency.



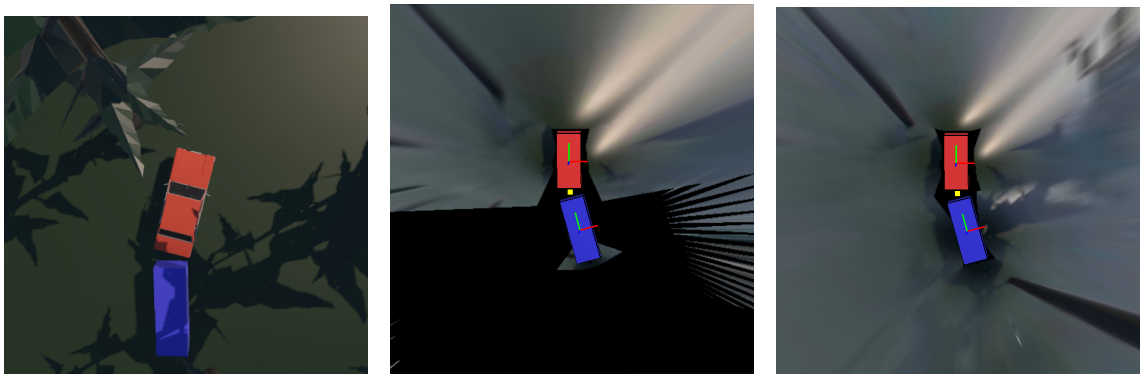
(a) True BEV. **(b)** Flat-plane BEV. **(c)** Tilted plane BEV.

Figure 4.3: Comparison of BEV projections using a flat-plane homography, the proposed tilted-plane homography, and the reference view while traversing a tilted road.

Table 4.2: Ground-truth orientation angles used in the simulated environment related to Figure 4.3.

Frame	**Roll [deg]**	**Pitch [deg]**	**Yaw [deg]**
Tractor frame	6.1245	4.5596	0.0000
Trailer frame	5.5999	-0.2071	33.8491

The subfigures in Figure 4.4 show the bird's-eye-view of the vehicle frames while ascending a hill. The flat-plane projection fails to project with IPM, while the tilted-plane projection maintains a more accurate position of terrain features, such as the adjacent trees.



(a) True BEV. **(b)** Flat-plane BEV. **(c)** Tilted plane BEV.

Figure 4.4: Comparison of BEV projections using a flat-plane homography, the proposed tilted-plane homography, and the reference view while traversing an off-road hill with adjacent trees.

Table 4.3: Ground-truth orientation angles used in the simulated environment, while traversing an off-road hill with adjacent trees.

Frame	Roll [deg]	Pitch [deg]	Yaw [deg]
Tractor frame	10.986900	-2.426951	0.0000
Trailer frame	8.842300	-5.557131	14.719390

4.2 Real-world scenario

The real-world scenario showed several notable differences compared to the simulation, primarily the larger scale and the presence of clear fisheye distortion, which mainly reduced the field of view after undistortion.

4.2.1 Camera pose estimation

The pose estimation using the current implementation showed promising results but suffered from performance issues, either with offset or high-jitter. The performance is visualized in the following plot in Figure 4.6 using IMU data as a reference. Note that while the IMU data is not ground truth, it is sufficiently reliable in the short term to evaluate the more volatile camera measurements.

Estimations from each camera pair showed huge amounts of uncertainty connected to the direction of the articulation angle, as visualized in Figure 4.5. Much of the noise was reduced through adaptive data selection, though not all of it could be eliminated. This algorithm has as mentioned, showed promising results and will be further discussed in Section 5.4.

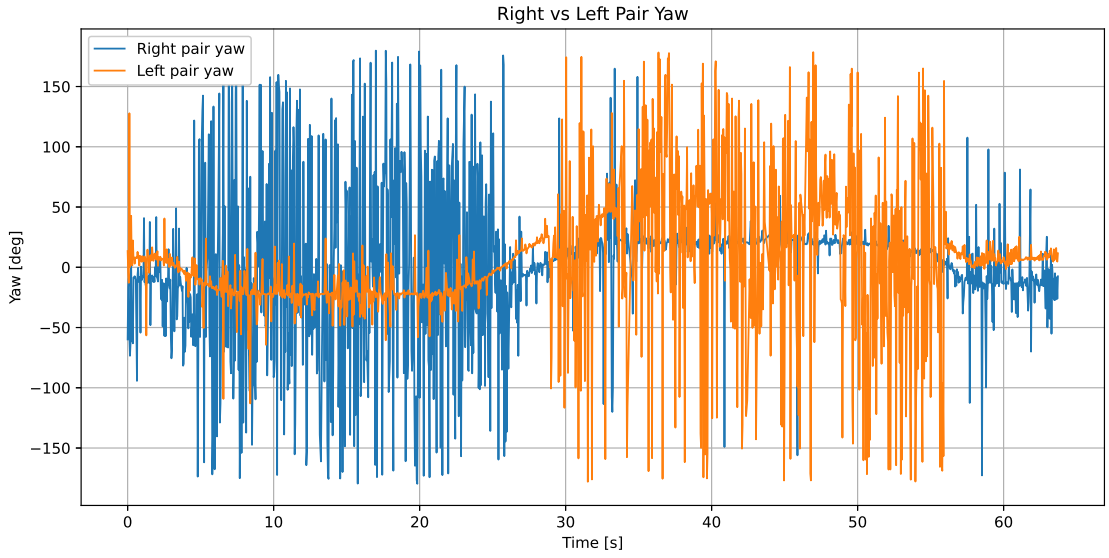


Figure 4.5: Individual measurements of the lens-to-lens rotation on each side.

A frame-by-frame run of the final visual pose estimation driving in a figure-8 pattern

showed the following yaw estimation, where much of the uncertainty is removed after processing, but still inheriting much of the jitter shown in the previous plot:

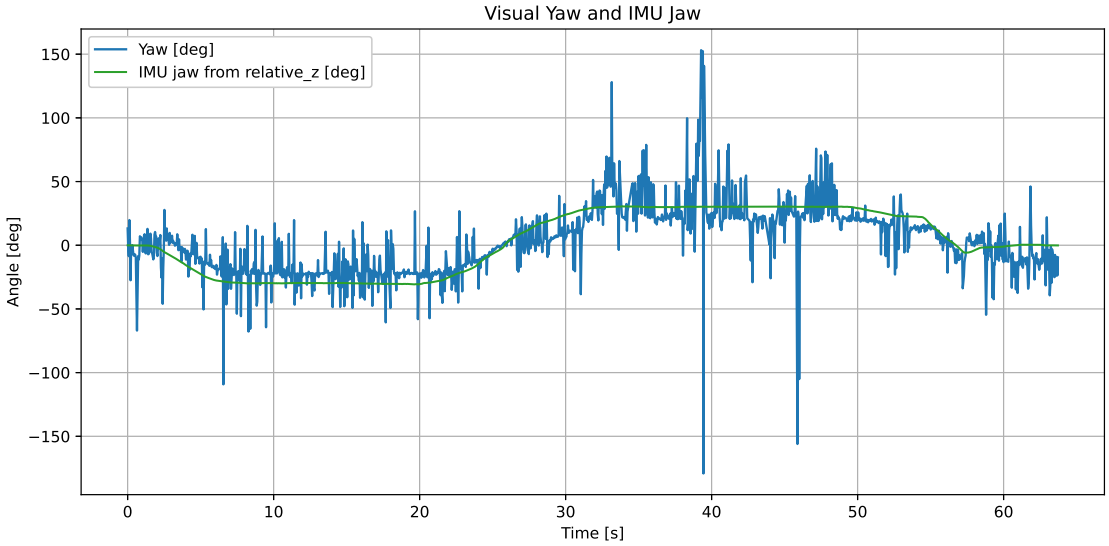


Figure 4.6: Result of visual pose estimation while driving in a figure-8 pattern.

4.2.2 Filtering

The filter uses every IMU update and fuses them with low-frequency camera samples marked with red dots in Figure 4.7.

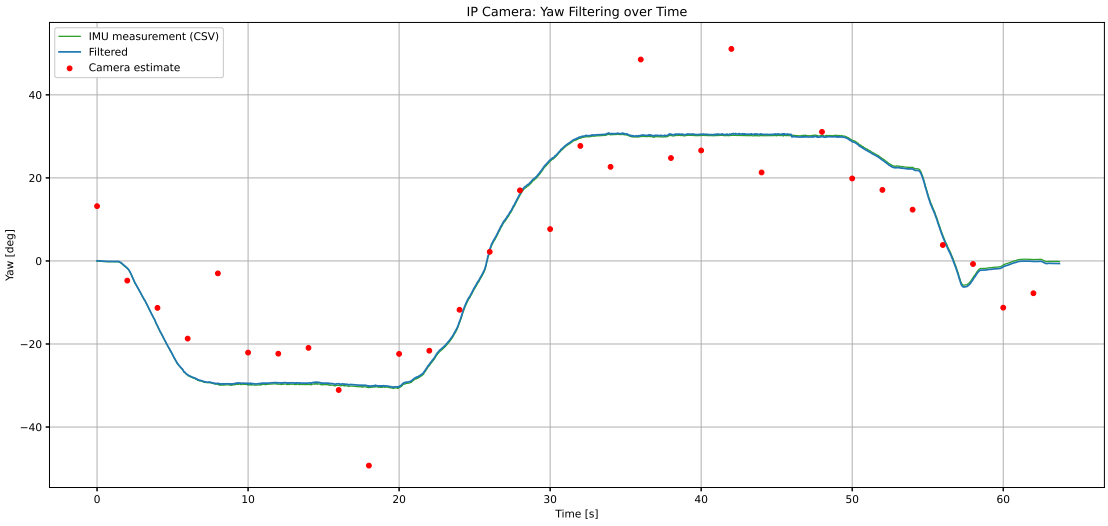


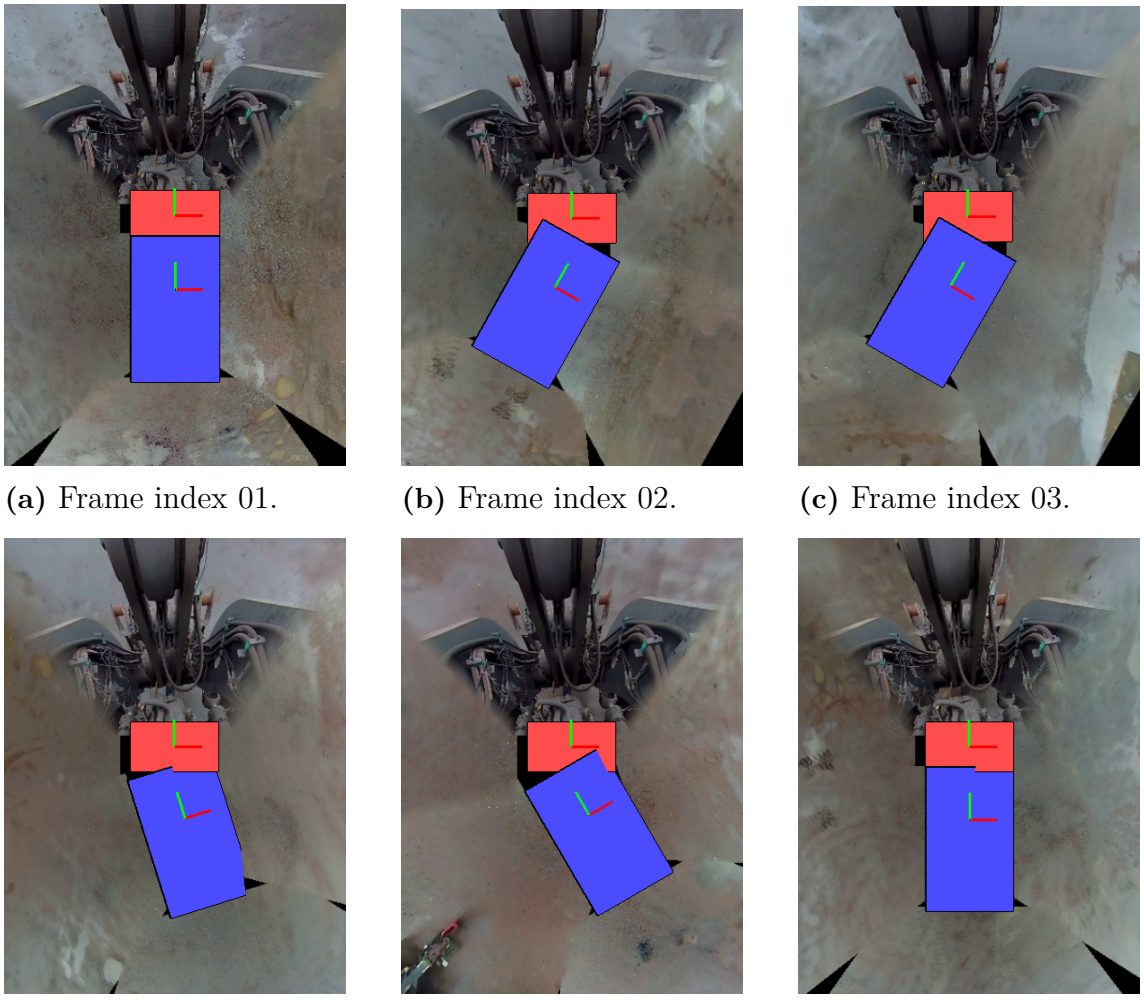
Figure 4.7: Filtering of the yaw angle while driving in a figure-8 pattern.

The camera measurements experience higher noise, which slightly limits their immediate utility within this system. However, they follow the trend of the actual

movement, which is the main point of using the camera data, i.e., for small adjustments of the yaw drift bias, countering the IMU drift during long runs. The reason for this lack in performance is discussed later in the report in Section 5.4. However, this result, together with the discussion, proves the validity of this system setup.

4.2.3 Bird's-eye-view

Figure 4.8 displays six frames from the test run, demonstrating the BEV performance during a figure-8 maneuver. The extrinsics are dynamically updated by using the data visualized in Figure 4.7. The front camera was mounted in a location that resulted in partial image obstructions from the wheel loader. In the figure, smooth transitions between the adjacent camera views indicate that the applied blending is sufficient to hide the otherwise hard image boundaries when there is moderate misalignment. However, there are minor visible intensity differences in overlapping regions, especially during sharp articulation.



(a) Frame index 01. **(b)** Frame index 02. **(c)** Frame index 03.

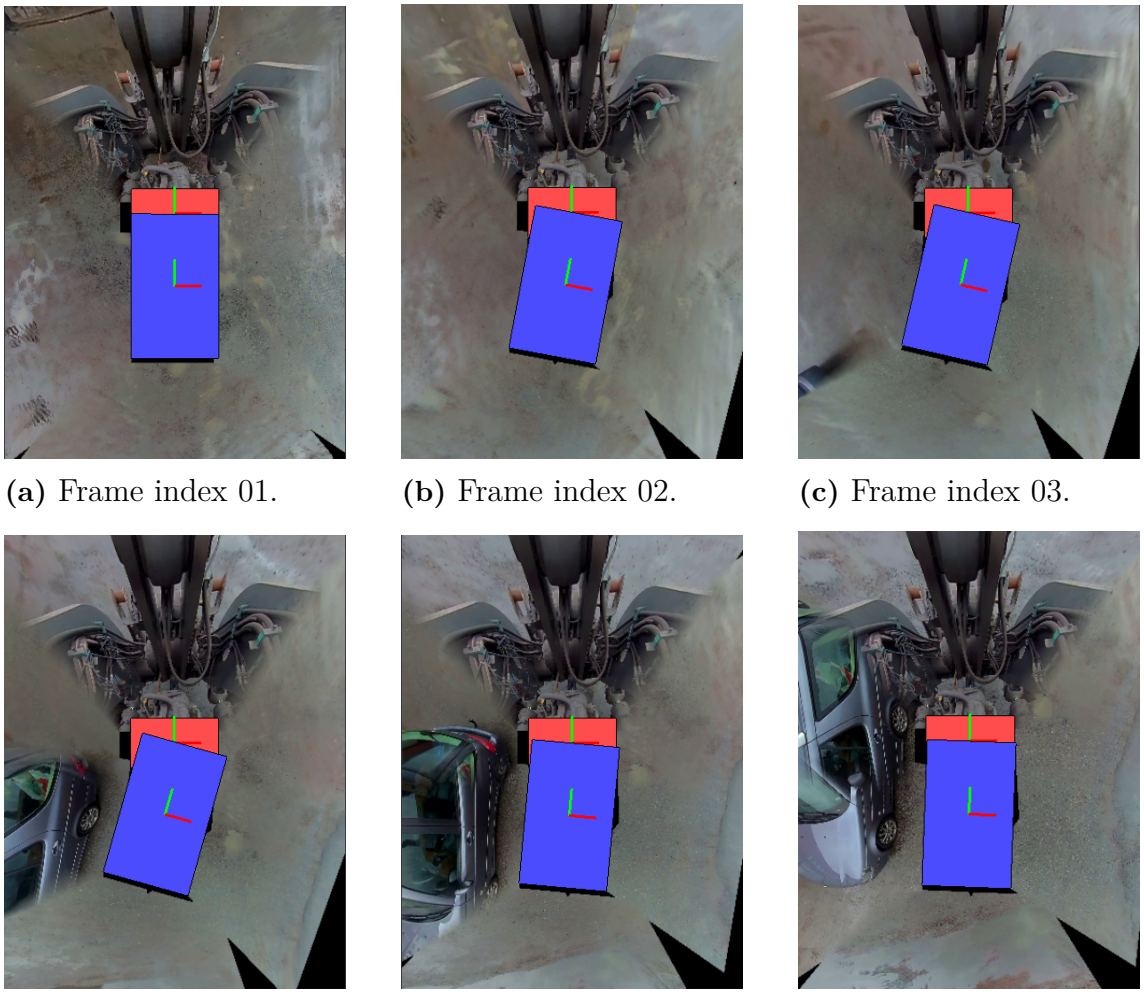
(d) Frame index 04. **(e)** Frame index 05. **(f)** Frame index 06.

Figure 4.8: Snapshots during driving in the figure-8 scenario.

Table 4.4: Vehicle and trailer orientation angles during the figure-8 scenario.

Frame index	Vehicle roll [deg]	Vehicle pitch [deg]	Vehicle yaw [deg]	Trailer roll [deg]	Trailer pitch [deg]	Trailer yaw [deg]
01	-0.9766	0.5078	0.0000	-0.7344	3.0313	-0.0625
02	-0.5625	1.2578	0.0000	-1.0391	3.3203	-29.2656
03	1.0625	1.6172	0.0000	0.3281	4.6016	-29.8438
04	0.9453	0.4766	0.0000	1.6016	2.8203	17.4688
05	-0.4609	0.3828	0.0000	-0.0547	3.0625	30.2031
06	0.8125	-0.3750	0.0000	1.0313	2.2578	-0.1563

An additional scenario was run where the wheel loader was reversed next to a car, the result of this scenario is visualized in Figure 4.9 and the table with the orientation data in table 4.5.



(a) Frame index 01. **(b)** Frame index 02. **(c)** Frame index 03.

(d) Frame index 04. **(e)** Frame index 05. **(f)** Frame index 06.

Figure 4.9: Snapshots of reversing scenario.

Table 4.5: Vehicle and trailer orientation angles during the reversing scenario.

Frame index	Vehicle roll [deg]	Vehicle pitch [deg]	Vehicle yaw [deg]	Trailer roll [deg]	Trailer pitch [deg]	Trailer yaw [deg]
01	0.8359	-0.2656	0.0000	1.0390	2.2256	-0.0039
02	1.5313	0.0703	0.0000	1.6481	2.8361	-10.5164
03	2.0156	-0.3594	0.0000	2.1097	2.5077	-12.9264
04	2.1328	0.5859	0.0000	1.9288	3.4063	-15.6087
05	2.1172	1.5156	0.0000	2.0364	4.1014	-5.1151
06	2.0781	1.1875	0.0000	2.0325	3.7580	-2.0788

4.3 Computational performance

The results from the computational performance evaluation were obtained through the methodology described in Section 3.6.2.

For the recorded data scenario, samples were collected with both laptops connected to power and set to performance mode. The results are presented in Table 4.6.

Table 4.6: Performance on recorded data.

Machine	Test 1 (fps)	Test 2 (fps)	Test 3 (fps)	Test 4 (fps)	Average (fps)
Lenovo Thinkpad	13.47	13.26	12.26	13.06	13.01
Dell Pro Max 18	15.23	14.91	14.83	14.79	14.94

In the real-time scenario, the laptop was running on battery and in performance mode and produced the following results in table 4.7.

Table 4.7: Performance on live data.

Machine	Test 1 (fps)	Test 2 (fps)	Test 3 (fps)	Test 4 (fps)	Average (fps)
Dell Pro Max 18	7.68	8.28	8.03	7.90	7.97

5
Discussion

This section will first present the topics of discussion from the methods and results of this thesis, thereafter the future works possible to build upon and improve as according to the authors.

5.1 Sensor setup

In hindsight, it can be concluded that the setup of sensors worked as intended when it came to what kind of information was both required and received. Ideally, some information about depth for better seam lines could perhaps have improved the results, requiring depth sensors, such as for example a LiDAR, or using the existing cameras for stereo vision. However, in practice, it is uncertain how much this information could actually improve the projection, given the limited research in the area of visual BEVs for other than flat homography projections.

During the research and planning stages of the project, an 8-camera setup where each camera would be placed in each corner of the two vehicle frames was discussed. This configuration could theoretically improve perception near the hitch, but substantial overlap across multiple views questions the necessity of additional cameras. The effects of increasing the camera count to 8 could possibly be replicated and give similar results by choosing cameras with a higher FOV for the setup used in this project, also reducing the cost and computational performance of the system as a whole. The front camera on the wheel-loader was, as mentioned in Section 4.2.3, partially obstructed. However, mounting it farther out on the vehicle frame was not feasible, as it contained mounting areas that were not rigid in comparison to the rest of the frame. If the camera configuration were increased from six to eight cameras, camera placements less limited to being in the front of the vehicle could have been explored to mitigate the obstruction problem.

It was mentioned in the related works Section 1.2 that one paper [16] utilizes an angle sensor for determining the articulation angle between the vehicle frames. This could also have been a competitive alternative to using IMUs placed on the vehicle frames. However, an angle sensor needs to be assembled at the hitch position, whereas IMUs can be placed almost arbitrarily on the vehicle body, making them more modular and versatile. Additionally, IMUs allow the system to easily measure the articulation in all three dimensions, which is more complicated to achieve with one or more angle sensors.

5.2 Intrinsic and extrinsic calibration

The intrinsic parameters were acquired using a ChArUco board and the algorithm described in Section 2.2.1 to obtain the intrinsic matrix and distortion coefficients. The same parameters were cached and used for all six cameras, which is a rough estimate since the intrinsics can be different even for different cameras of the same type. However, what probably made the largest difference in terms of the real-world results was the camera extrinsics. The used approach on the real-world setup, where everything was measured by hand, is a coarse method. Even small deviations in both transformation and rotation may give significantly inaccurate extrinsic matrices. The reason for this method is that it is a separate and not a core area of research for this thesis. It might have caused the project to not reach the intended results, able to answer the research questions, while instead focusing on an already explored area of research, as that conducted for example by Li et al. [16].

However, if a more accurate calibration method, both intrinsically and extrinsically, were to be incorporated, the results can be assumed to improve in several ways. One of them being better alignment in the final surround-view projection between visible objects in the images, as well as disappearing and duplicated objects. The results from the currently developed BEV in the Unity environment gives the closest to what good calibration would achieve, since it uses cameras with a larger horizontal and vertical FOV than the real-world cameras, together with no fisheye lens distortions as the cameras are setup as using the pinhole approach with analytically known values for the used camera model.

5.3 Filtering and measurement usage

The reasoning behind using a Kalman filter was first and foremost to counter the drift that appears in the yaw measurements from the IMUs. Filtering received limited priority for the minor noise in pitch and roll measurements, resulting in inconsistent filtering across states. The effect of the filter on trailer pitch and roll was minimal and could have, in hindsight have been removed from the state vector, simplifying the filtering process, especially as the values themselves are already filtered internally in the IMUs. The measurements that were passed through the filter before usage are presented here:

$\varphi_{w \to tractor} \rightarrow$ Surround-view
$\theta_{w \to tractor} \rightarrow$ Surround-view
$\psi_{w \to tractor} \rightarrow$ Filter $\rightarrow$ Surround-view
$\varphi_{w \to trailer} \rightarrow$ Filter $\rightarrow$ Surround-view
$\theta_{w \to trailer} \rightarrow$ Filter $\rightarrow$ Surround-view
$\psi_{w \to trailer} \rightarrow$ Filter $\rightarrow$ Surround-view

A justified approach could instead have been to use a simplified Kalman filter without the relative pitch and roll values, which would have cut the number of states in

more than half. Instead of filtering the pitch and roll, perhaps a separate low-pass filter could have been used on these measurements to counter some of the noise across those system states.

Nevertheless, as the result proved promising from the existing setup, no improvement in quality is expected from a better method in reducing orientation noise. Conversely, it could have a minor effect on computational performance, as reducing the Kalman state vector and applying simple low-pass filters would decrease the operations per timestep.

5.4 Visual orientation estimation performance

The reason behind having a visual pose estimator was as described earlier in the report, which is to complement the IMU in long-term estimations, where drift could be a factor, causing the IMU to output unreliable data in the yaw-axis. The cameras are then supposed to reliably correct the measurements using a built-in yaw bias in the Kalman filter. Throughout the project, it suffered from performance issues, most likely stemming from mainly two causes: a lack of features in its FOV and bad intrinsic and extrinsic calibration, which both can be proven to cause bad data.

The case of bad intrinsic and extrinsic calibration was increasingly prevalent when moving from the simulated data in Unity to the real-world case with manually calibrated fisheye cameras. In Unity with known calibration, the estimator was able to follow the real values without offset and with quite high accuracy, as shown in Figure 5.1. This was probably due to the fact that calibration was more exact on the simulated vehicle, together with the fact that the cameras were slightly more angled up, meaning more features in the image field, giving better conditions for RANSAC.

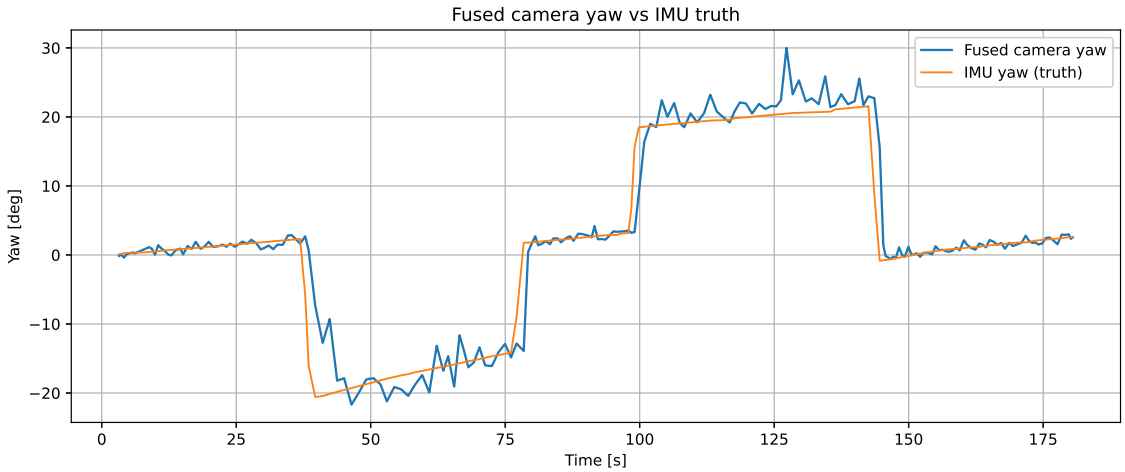


Figure 5.1: Pose estimation when using only cameras in Unity.

However, when using real data, a clear offset appears as presented in the results in Figure 4.6 and even more clearly for each side in Figure 4.5, where we can see the

difference in offset from each side.

A probable cause for this change is that the cameras, in the real-world scenario, are not calibrated accurately enough. In Unity, the exact location of the cameras are known, whereas they have to be accurately measured and calibrated in the real world. For the cameras used in this project, this is especially difficult as the lens was loose and could be arbitrarily angled out of the camera housing. Algorithmic extrinsic calibration has been out of scope for this project, and the result of doing this correctly can only be discussed.

The intrinsic calibration also comes into play when moving to a real-world scenario, as distortion increases when moving to real 150-degree fisheye lenses. This means that it is increasingly important to know how the frames are distorted to know how the features near the edges relates to each other. Multiple attempts were made to correctly calibrate the cameras, but finding the optimal camera matrix and distortion coefficients was difficult and evaluating the results was challenging.

The second reason for bad performance in the visual pose estimation is the lack of good features in the image frames. The cameras on the vehicle frame are angled down towards the ground, which means that when driving on gravel, few features can be matched as seen in Figure 5.2, thus causing an inaccurate estimation of the transformation between the perspectives.

Figure 5.2: View of the left and right side camera pair of the vehicle.

To counter this, the optimal solution would be to first angle the cameras further up, allowing them to see more of the surrounding and off-ground features. This would, however, require higher FOV cameras than used in this project, as not to sacrifice image data close to the vehicle. Second, different algorithms for detecting features could be tested. This would require a study regarding using feature detection on ground-facing and low-texture images.

5.5 Evaluation of tilted ground plane assumptions

The tilting ground plane approach was one of the thesis's most central hypotheses to try to improve the results when the ground is not entirely flat and IPM suffers. Even though the currently implemented tilting ground planes do not replace the potential improvements that dense depth data could achieve as previously discussed in Subsection 5.1, it helps to mitigate some of the otherwise incorrect projection effects when trying to project onto a tilted surface such as a hill or mountain. However, the topography of a mountain is seldom able to be fully represented by two changing planes. This approach is a first-order approximation that works to better understand the terrain while not sacrificing speed, because of the computational efficiency by using an IMU together with a camera to get reliable orientation data for the vehicle frames.

Comparing the results of the flat and tilted plane approaches in Figures 4.3b and 4.3c, or Figures 4.4b and 4.4c, reveals distinct projection behaviors. One visual conclusion to draw is that the camera projections get inverted if below the ground plane, which they are in the flat-plane figures, since it uses one common plane defined from the tractor vehicle. This is a known geometric consequence of IPM, since IPM implicitly assumes all rays intersect the plane in front of the camera. However, the tilted plane approach is not universally favorable. As shown in Figure 4.2a compared to Figure 4.2c, road markings exhibit ghosting artifacts in the tilted projection. This could be a result of a non-negligible tilt in roll for the trailer frame compared to the tractor frame, as can be interpreted from Table 4.1. A solution could be to have a dynamic method changing between the flat and tilted techniques when the amount of tilt in either pitch or roll for either the tractor or trailer vehicle frame exceeds a certain threshold. This would allow the use of both the strengths of the non-tilted approach, which allows for a more consistent world ground plane in planar situations. The tilted approach would be used in settings such as off-road or mountainous roads, which would improve the stitching quality and the interpretable information given from the BEV.

It is again important to note that this approach does not aim to reconstruct dense three-dimensional geometry of the environment. The tilting planes still uses IPM, which only gives fully accurate projections on flat terrain. Likewise, as previous bird's-eye-view approaches for ADAS systems, non-planar objects, such as rocks, adjacent cars, trees, get stretched in the projections as can be seen in Figure 4.4c. This behavior is consistent with prior BEV-based ADAS systems and reflects a fundamental limitation of planar projection rather than an implementation error.

Most existing research addressing such distortions does not attempt to refine the projected image itself, but instead, the current state-of-the-art research leans more towards bird's-eye-view systems, giving semantic and object data to be used for decision-making for autonomous vehicles instead of a driver-assisting system. While effective for autonomous decision-making, these approaches move away from visually interpretable representations intended for driver assistance. In contrast, this

work maintains a visual BEV representation and therefore must explicitly address the geometric inconsistencies that arise from camera overlap, imperfect calibration, and parallax effects when objects appear in multiple views. In practice, these inconsistencies are seen as duplicated objects, unstable stitching boundaries, or objects appearing multiple times in the BEV. These effects are especially prevalent in articulated vehicles, where camera viewpoints change dynamically due to articulation. An object may not only be duplicated, but can also be viewed from different angles and showing different sides of that same object. Since the object has 3D-dimensional data, it will be stretched and distorted in the bird's-eye-view. These objects will either intersect or be angled differently, which can be distracting or even deceitful.

5.6 BEV image processing techniques

Other image processing techniques directly related to improving the quality of the BEV have been image undistortion, adaptive seam alignment and feather blending.

5.6.1 Image undistortion

The undistortion effect is possible to see as one of the incorporated results from the subfigures in Figure 4.8. The technique itself is part of OpenCV, it requires good distortion coefficients for good performance. If the initial camera calibration had been improved, the fisheye calibration could have been assumed to be better rectified, and then give a better projection in the BEV.

5.6.2 Adaptive seam alignment

Unlike rigid vehicles, where seam lines do not require adjustment and can remain fixed, this approach is insufficient for articulated vehicles, as it may assign critical regions to cameras observing them from unfavorable angles. By instead defining the seam directions of adjacent cameras projected onto the ground plane, the seam alignment adapts dynamically to the current articulation angle. This approach favors the camera projection that is geometrically closer to the observed scene region, thereby maximizing pixel density and reducing perspective distortion in object projections.

However, this is still a method achieving the best results with a flat ground and without adjacent objects. This is not a result of the seam alignment used, but as previously mentioned, a consequence of this projection method. In image stitching, dynamic seam stitching is often cited to reduce parallax and duplicate objects. That method is though, best suited for panorama images, where the images are not meant to be projected onto a flat ground projection but instead stitched side by side, without the risk for ground-plane projection error.

5.6.3 Feather blending

Feather blending was employed to smooth transitions in overlapping regions of the projected camera images. By assigning higher blending weights to pixels farther from image boundaries, the method reduces visible seam artifacts that would otherwise arise from abrupt transitions between camera views. Using blending, however, does not fix bad geometric inconsistencies, as observed in the tilted-plane experiment results, they can not fully compensate for misalignment caused by differing plane orientations.

5.7 Composed final BEV

As per the subfigures in 4.8, it is possible to see a fully projected BEV during a figure-eight maneuver in a real-world environment. The BEV is generated by dynamically updating the camera extrinsics based on the individually estimated vehicle and trailer orientations.

Visually, there are black non-textured areas because of cameras with not enough field of view to create enough overlap for all of the projections. There is also an obstruction taking up most of the front camera, as a consequence already described in Section 4.2.3. This demonstrates that the projection pipeline dynamically updates extrinsics and adjusts the output as articulation changes. The results shows that a BEV for an articulated vehicle using a VINS-based orientation method can be achieved, while using a tilting-ground plane method together with IPM.

5.7.1 Performance discussion

In Section 4.2.3, the computational performance averaged 13.01 fps for the Lenovo Thinkpad and 14.94 fps for the Dell Pro Max 18. Despite the Dell Pro Max 18 providing substantially higher theoretical computational performance, the observed difference in frame rate was relatively small. This behavior suggests that the current implementation is primarily CPU-bound and not extensively optimized for parallel execution or GPU acceleration. While the measured frame rates are below typical video refresh rates, they are sufficient for almost real-time feedback to a driver using the system.

5.8 Future works

To further develop the project and improve its results, several areas have been identified throughout the project that could be of interest to investigate. Many of these paths have a high probability of having a pronounced effect on the result.

5.8.1 Higher FOV-cameras

It has been previously mentioned during the report that the cameras had too low FOV. When constructing surround-view systems for automotive applications, cam-

eras typically have a field of view exceeding 180° [29, 30, 31]. This means that there is a lot more margin in the overlap, minimizing the risk of dark areas where there is a lack of coverage.

It is highly likely that introducing higher FOV-cameras than those used during this thesis might result in a higher quality result and improved vision closer to the vehicle for the operator.

5.8.2 Improved intrinsic and initial extrinsic calibration

As mentioned throughout the report and previously in the discussion, the lack of accurate extrinsic and intrinsic calibration of the real cameras negatively affected the result, both in the visual pose estimation and in the final BEV-projection.

It would therefore be interesting as part of future work to use well-established extrinsic calibration methods to obtain more accurate positional and rotational data of the cameras, to investigate how much increase in performance and quality can be obtained in the process areas currently suffering from calibration issues.

The same goes for the intrinsic calibration of the camera parameters. The process of determining those in this project was conducted simply and more effort and better calibration methods could be applied for better distortion rectification in the image and its edges.

5.8.3 Feature detection and matching techniques

Also previously mentioned in the results and discussion, the performance of the visual orientation estimator has had suboptimal performance. This issue can partly be attributed to inaccurate extrinsic calibration. However, another integral part of performing feature matching is choosing the right model for the detector and the matcher. This project has used the baseline models during its development, SIFT and ORB, in combination with brute force and KNN matchers. An interesting continuation of this project would be to investigate what kinds of models and combinations between detectors and matchers are the most optimal for the use-case of articulated vehicles in a varied environment.

A lot of research has been done in the area of feature-based orientation estimation and localization. An example is the previously cited paper by Dong et al. [12] who uses the SuperGlue and SuperPoint algorithms [32], [33] for feature detection and matching. This is more resource demanding, but an interesting question to look into would be its comparative speed and if it would be feasible running online on existing hardware.

Other papers try to use an ORB/SURF detector and FLANN matcher [34] to solve the issue, pointing out that several very different approaches can be used as well as combinations between existing algorithms. Finding the correct combination of algo-

rithms for the use-case of this problem could therefore be an interesting progression.

5.8.4 Parallelization and GPU-based processing

The current implementation primarily executes on the CPU with limited parallelization. Several parts of the BEV creation pipeline are, however, well-suited for parallel and GPU-based execution. In particular, camera image warping, homography application, mask generation, and image blending operate on a per-pixel basis and could be efficiently parallelized.

By utilizing, for example, CUDA or OpenCL, there are significant performance improvements to be expected. These optimizations are expected to reduce latency and enable higher frame rates, making the system more suitable for fully real-time driving and operation.

5.8.5 Object-level and semantic BEV

Several other recent works have shown the benefits of semantic and object-level bird's-eye view representations for scene understanding. DaF-BEVSeg proposes a distortion-aware BEV segmentation framework, producing a semantically meaningful top-down representation that emphasizes objects and structural elements rather than dense image texture[35]. This approach emphasizes objects relevant for the driver's understanding, such as vehicles, obstacles and drivable space.

A similar concept could be integrated into the currently proposed BEV pipeline as a driver-assistance feature, with the aim of improving situational awareness and reducing visual clutter. Such a representation would not be intended for autonomous decision-making, but rather as another helpful layer to support the driver during complex maneuvers, particularly for articulated vehicles.

6
Conclusion

This master's thesis has explored the feasibility of a dynamic surround-view system for articulated vehicles by combining camera-based image processing with inertial measurements from IMUs. The work was motivated by the limitations of existing conventional bird's-eye-view (BEV) systems, which often assume a rigid vehicle frame and constant even terrain, which is not a scenario that always holds for articulated vehicles.

A full pipeline was developed, including sensor data, pose estimation through sensor fusion and homography-based BEV generation. A key contribution of this work is the introduction of independently tilted ground planes for each vehicle frame, allowing the projection to adapt to full three-dimensional articulation on uneven terrain.

Real-world experiments confirmed the overall feasibility of the system design. While the lack of high-quality extrinsic calibration affected the result, both in terms of the visual pose estimation and the final projected BEV, the general system performance showed successful and promising results and works well as a proof of concept.

The scope of this work focused on computational efficiency, robustness, and ease of interpretation rather than detailed three-dimensional reconstruction. Consequently, the BEV is based on a planar approximation and therefore exhibits known limitations of inverse perspective mapping, such as distortion of objects above the ground and artifacts caused by parallax. These effects are well known in existing BEV-based driver assistance systems and stem from inherent limitations of planar projection rather than from the proposed method itself.

In summary, this thesis demonstrates that IMU-camera fusion combined with per-frame tilted ground planes provides a viable and efficient approach for extending surround-view systems to articulated vehicles while addressing uneven terrain challenges. Future work should focus on improved camera calibration, more robust visual feature extraction, further testing on full-scale articulated vehicles with three-dimensional articulation on tilting ground planes, and if object-level and semantic data could be integrated to further improve the driver's scene understanding.

Bibliography

[1] L. Thodis, "White car on parking lot in bird's eye view." https://www.pexels.com/photo/white-car-on-parking-lot-in-birds-eye-view-19577721/, 2026. Photograph, Pexels.

[2] Millsy, "Caterpillar 740 ejector articulated haul truck." Wikimedia Commons, 2005. Public domain image.

[3] F. Schwichtenberg, "Terex tl260 baggerlader at hamburg airport." https://commons.wikimedia.org/wiki/File:Terex_TL260_Baggerlader_Hamburg_Airport_RHM_01_01.jpg, Mar. 2014. Wikimedia Commons, licensed under Creative Commons and/or GFDL.

[4] M. Meier, "Pictorial: Articulated bus." https://inchbyinch.de/pictorial/articulated-bus/, 2020. Creative Commons Attribution–NonCommercial (CC BY-NC). Accessed: 2026-04-29.

[5] CPAC Systems, "Imu product page – we are cpac systems," 2026. Accessed: 2026-04-01.

[6] Global Market Insights, "Automotive surround view system market size & share, 2024–2032." https://www.gminsights.com/industry-analysis/automotive-surround-view-system-market, 2024.

[7] MathWorks, "Create 360° bird's-eye-view image around a vehicle." https://www.mathworks.com/help/driving/ug/create-360-birds-eye-view-image.html, 2026. MATLAB & Simulink Documentation.

[8] K. Eckenhoff, P. Geneva, and G. Huang, "Mimc-vins: A versatile and resilient multi-imu multi-camera visual-inertial navigation system," *IEEE Transactions on Robotics*, vol. 37, no. 5, pp. 1360–1380, 2021.

[9] V. R. Kumar, C. Eising, C. Witt, and S. K. Yogamani, "Surround-view fisheye camera perception for automated driving: Overview, survey & challenges," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 4, pp. 3638–3659, 2023.

[10] N. Markó, Z. Rózsa, Á. Ballagi, and T. Szirányi, "Robust road surface normal and pitch prediction via imu-camera fusion," in *Advanced Concepts for Intelligent Vision Systems (ACIVS 2025)*, vol. 15656 of *Lecture Notes in Computer Science*, pp. 591–603, Springer, 2026.

[11] G. Huang, "Visual-inertial navigation: A concise review," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 9572–9582, 2019.

[12] Z. Dong, M. Fu, H. Liang, C. Zhu, and Y. Yang, "DSVT: Dynamic 3d surround view for tractor-trailer vehicles based on real-time pose estimation with drop

model," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 9461–9467, 2024.

[13] L. Sun, H. Liang, Z. Dong, Y. Yang, and M. Fu, "Udsv: Unsupervised deep stitching for tractor-trailer surround view," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5157–5163, 2025.

[14] H. Wang, Q. Zhang, Z. Zheng, X. Li, H. Tan, and R. Li, "A low-texture robust hybrid feature based visual odometry," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 10113–10120, 2024.

[15] A. Minervini, A. Carrio, and G. Guglieri, "Enhancing visual–inertial odometry robustness and accuracy in challenging environments," *Robotics*, vol. 14, no. 6, 2025.

[16] W. Li, L. Cao, P. Yue, S. Wen, J. Liao, J. Xia, and X. Feng, "Vavm: A flexible technique for variable-angle around view monitor system towards articulated engineering vehicle," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 4, pp. 3556–3568, 2022.

[17] B. L. Song, "Tour eiffel – wide angle." `https://commons.wikimedia.org/wiki/File:Tour_Eiffel_Wikimedia_Commons.jpg`, 2009. Image licensed under CC BY-SA 3.0.

[18] "Eiffel tower, paris, france." `https://www.needpix.com/photo/77697/eiffel-tower-paris-france-seine-river-landmark-french-urban`, 2022. Public domain image from Needpix; no attribution required.

[19] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2nd ed., 2004.

[20] O. Team, "OpenCV: Open source computer vision library," 2024. Accessed: 2026-04-29.

[21] Wikipedia contributors, "Kalman filter," 2026. Accessed: 2026-03-26.

[22] CPAC Systems AB, "Imu datasheet." Technical Datasheet, 2024. Accessed: 2025-03-25.

[23] STONKAM Co., Ltd., "Fhd410n IP camera." `https://www.stonkam.com/products/IP-Camera-FHD410N.html`, 2026. Accessed: 2026-04-04.

[24] "Cartoon Race Track Oval." *Unity Asset Store*, n.d. [Online]. Available: https://assetstore.unity.com/packages/3d/environments/roadways/cartoon-race-track-oval-175061.

[25] "Low-Poly 1970s Family Sedan 3D Model." *Unity Asset Store*, n.d. [Online]. Available: https://assetstore.unity.com/packages/3d/vehicles/land/low-poly-1970s-family-sedan-3d-model-free-download-car02-343310.

[26] "GStreamer Multimedia Framework." *GStreamer Project*, n.d. [Online]. Available: https://gstreamer.freedesktop.org/.

[27] R. Hartley, J. Trumpf, Y. Dai, and H. Li, "Rotation averaging," *International Journal of Computer Vision*, vol. 103, no. 3, pp. 267–305, 2013.

[28] R. Szeliski, *Image Alignment and Stitching: A Tutorial*, vol. 2.1 of *Foundations and Trends in Computer Graphics and Vision*. Now Publishers, 2006.

[29] J. Cho, J. Lee, J. Ha, P. Resende, B. Bradai, and K. Jo, "Surround-view fisheye camera viewpoint augmentation for image semantic segmentation," *IEEE Access*, vol. 11, pp. 48480–48492, 2023.

[30] Robert Bosch GmbH, "360 degree surround view." Online, n.d. Accessed: 24 April 2026.

[31] V. Appia, H. Hariyani, S. Sivasankaran, S. Liu, K. Chitnis, M. Mueller, U. Batur, and G. Agarwal, "Surround view camera systems for adas." White paper, 2016. Accessed: 24 April 2026.

[32] Hugging Face, "Superglue." `https://huggingface.co/docs/transformers/model_doc/superglue`, 2024. Accessed: 29 Apr. 2026.

[33] Hugging Face, "Superpoint." `https://huggingface.co/docs/transformers/model_doc/superpoint`, 2024. Accessed: 29 Apr. 2026.

[34] Y. Yu, J. Ni, X. Wang, and J. Tian, "Optimization of orb feature point detection algorithm for image matching application," in *2024 7th International Conference on Mechatronics and Computer Technology Engineering (MCTE)*, pp. 1519–1523, 2024.

[35] S. Yogamani, D. Unger, V. Narayanan, and V. R. Kumar, "DaF-BEVSeg: Distortion-aware fisheye camera based bird's eye view segmentation with occlusion reasoning," *arXiv preprint arXiv:2404.06352*, 2024.

A

Extrinsics and intrinsics for the simulation and real world environment

A.1 Simulation

The articulated vehicle model consists of a tractor frame, a trailer frame, and a hitch joint. The reference world frame is defined at the origin of the tractor frame.

- Tractor frame origin: $\mathbf{o}_{\text{vehicle}} = [0.0,\ 0.0,\ 0.0]^{\mathsf{T}}$
- Hitch joint position relative to the tractor frame:

$$\mathbf{t}_{\text{vehicle}\to\text{hitch}} = \begin{bmatrix} 0.0 \\ -2.6779 \\ 0.0 \end{bmatrix}$$

- Trailer frame origin relative to the hitch:

$$\mathbf{t}_{\text{hitch}\to\text{trailer}} = \begin{bmatrix} 0.0 \\ -2.3221 \\ 0.0 \end{bmatrix}$$

A.1.1 Camera positions

Six cameras are mounted on the tractor vehicle and trailer frames, three on each vehicle frame. All camera positions are defined relative to their respective parent frames, where p_1, p_2 and p_6 adheres to the tractor vehicle and p_3, p_4 and p_5 to the trailer frame. The measurements are expressed in meters.

Table A.1: Camera mounting yaw angles

Camera	Yaw angle ψ_c [deg]
1	0
2	−90
3	−90
4	−180
5	90
6	90

$$\mathbf{p}_1 = \begin{bmatrix} 0.0 \\ 2.1 \\ 0.5630 \end{bmatrix} \qquad \mathbf{p}_2 = \begin{bmatrix} 1.0 \\ 0.0 \\ 0.5330 \end{bmatrix}$$

$$\mathbf{p}_3 = \begin{bmatrix} 0.6979 \\ 0.0 \\ 0.5300 \end{bmatrix} \qquad \mathbf{p}_4 = \begin{bmatrix} 0.0 \\ -2.0369 \\ 0.5300 \end{bmatrix}$$

$$\mathbf{p}_5 = \begin{bmatrix} -0.6979 \\ 0.0 \\ 0.5300 \end{bmatrix} \qquad \mathbf{p}_6 = \begin{bmatrix} -1.0 \\ 0.0 \\ 0.6130 \end{bmatrix}$$

A.1.2 Camera orientations

Each camera orientation is defined using Euler rotations expressed in a ZYX convention. All cameras share the same downward pitch of $-25°$ to orient the field of view toward the ground plane. An additional yaw rotation aligns each camera with its intended viewing direction.
The rotation matrix for each camera is computed as

$$\mathbf{R}_c = \mathbf{R}_z(\psi_c)\,\mathbf{R}_x(-90°)\,\mathbf{R}_x(-25°)$$

where ψ_c denotes the camera-specific yaw angle.

The extrinsic parameters defined in this appendix are used as fixed input to the BEV generation pipeline. During runtime, the effective camera poses are updated by combining these static camera extrinsics with dynamically estimated vehicle and trailer orientations.

A.1.3 Camera intrinsics

The following camera matrix was used for the simulation environment:

$$\mathbf{K}_{unity} = \begin{bmatrix} 64.3078 & 0 & 240 \\ 0 & 64.3078 & 240 \\ 0 & 0 & 1 \end{bmatrix}$$

As the camera was virtual, there was no distortion, thus no distortion coefficients were defined.

A.2 Real world

The articulated test vehicle consists of a tractor frame, a trailer frame, and a revolute hitch joint connecting the two. All dimensions are expressed in meters.
The hitch joint is defined relative to the tractor frame origin as

$$\mathbf{t}_{\text{vehicle}\to\text{hitch}} = \begin{bmatrix} 0.0 \\ -1.0 \\ 0.0 \end{bmatrix}.$$

The trailer frame origin is defined relative to the hitch joint as

$$\mathbf{t}_{\text{hitch}\to\text{trailer}} = \begin{bmatrix} 0.0 \\ -0.92 \\ 0.0 \end{bmatrix}.$$

This kinematic structure is identical to that used in the simulation environment, allowing the same transformation chain to be applied consistently in both simulated and real-world experiments.

A.2.1 Camera positions

Six cameras are mounted on the tractor and trailer frames. Camera positions are defined relative to their respective parent frames and expressed in meters.

$$\mathbf{p}_1 = \begin{bmatrix} 0.0 \\ 0.0 \\ 2.1 \end{bmatrix} \qquad \mathbf{p}_2 = \begin{bmatrix} 1.15 \\ 0.0 \\ 2.1 \end{bmatrix}$$

$$\mathbf{p}_3 = \begin{bmatrix} 1.2 \\ 0.0 \\ 1.9 \end{bmatrix} \qquad \mathbf{p}_4 = \begin{bmatrix} 0.0 \\ -2.36 \\ 2.05 \end{bmatrix}$$

$$\mathbf{p}_5 = \begin{bmatrix} -1.2 \\ 0.0 \\ 1.9 \end{bmatrix} \qquad \mathbf{p}_6 = \begin{bmatrix} -1.15 \\ 0.0 \\ 2.1 \end{bmatrix}$$

The front, right-front, and left-front cameras are mounted on the tractor frame, while the remaining cameras are mounted on the trailer frame.

A.2.2 Camera orientations

Camera orientations are defined using a ZYX Euler angle convention. All cameras are rotated downward to observe the ground plane and yawed to cover the surroundings of the vehicle. The rotation matrix of each camera is constructed as

$$\mathbf{R}_c = \mathbf{R}_z(\psi_c)\,\mathbf{R}_x(-90^\circ)\,\mathbf{R}_x(\alpha_c),$$

Table A.2: Camera orientations on full scale wheel loader

Camera	Yaw ψ_c [deg]	Tilt α_c [deg]
1	0	-45.0
2	-90	-57.5
3	-90	-59.5
4	-180	-56.0
5	90	-55.0
6t	90	-45.4

where ψ_c is the yaw angle determining the viewing direction and α_c is the camera-specific downward tilt angle.
The parametric camera poses listed in this appendix are equivalent to standard 3×4 extrinsic matrices and are converted into matrix form internally during homography computation and BEV generation.

A.2.3 Camera intrinsics

The following camera matrix was used for the simulation enviroment:

$$\mathbf{K} = \begin{bmatrix} 730.0 & 0.0 & 1004.33506 \\ 0.0 & 730.0 & 534.881456 \\ 0.0 & 0.0 & 1.0 \end{bmatrix}$$

And the following distortion coefficients was used as:

$$\mathbf{D} = \begin{bmatrix} -0.03698064 & 0.00851613 & -0.01319105 & 0.00338194 \end{bmatrix}$$

DEPARTMENT OF MECHANIC AND MARITIME SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY