University of Tartu

Faculty of Science and Technology

Institute of Technology

Jürgen Leppsalu

Video Compression Effect on a Camera-based Occupancy Prediction Model

Master's Thesis (30 ECTS)

Supervisor:

MSc Rando Avarmaa

Tartu 2025

Resümee/Abstract

Video kompressiooni mõju kaamerapõhisele hõivatuse ennustussüsteemile

Kaamerapõhised hõivatuse ennustusmudelid hindavad takistuste olemasolu ja kaugust üksnes kaamera piltide ja positsioneerimisandmete põhjal. Käesolevas lõputöös uuritakse, kuidas mõjutab video kompressioon sellise mudeli käitumist. Selle uurimiseks rakendati video kompressiooni kolme koodekiga: MJPEG, H.264 ja H.265. Kompressitud kaadritega treeniti 15 hõivatuse ennustusmudeli versiooni ning analüüsiti valideerimiskao muutust iga konfiguratsiooni puhul. Tulemused näitavad, et valitud hõivatusennustusmudel on kompressiooni artefaktide suhtes vastupidav. Kuigi agressiivsed kompressiooni variandid põhjustasid järjestikuste kaadrite puhul vähem stabiilseid ennustusi, siis jäid paljud struktuurid ikkagi ennustustest nähtavaks. Tulevased tööd peaksid uurima, kas ka pärismaailma andmestikel treenitud mudelid näitavad sarnast vastupidavust, kuna käesolev töö käsitleb kompressiooni mõju CARLA simuleerimiskeskkonnas genereeritud sünteetiliste andmete põhjal.

CERCS: P176 Tehisintellekt; T111 Pilditehnika.

Märksõnad: masin nägemine, kauguse ennustamine, video kompressioon, hõivatuse ennustusmudel

Video compression effect on a camera-based occupancy prediction model

Camera-based occupancy prediction models estimate the presence and distance of obstacles using camera images and camera positional data. This thesis investigates how lossy video compression impacts the performance of such a model. To test this, video compression was applied using three video codecs: MJPEG, H.264, and H.265. The compressed frames were then used to train 15 versions of an occupancy prediction model, and the resulting validation loss was analyzed for each configuration. The findings show that the selected occupancy prediction model demonstrated a high degree of resilience to compression artefacts. Although high-compression variants produced less stable predictions across consecutive frames, the predictions still contained mostly valid structures that resembled the expected outcome. As this study was conducted using synthetic data generated in the CARLA simulation environment, future research should explore whether models trained on real-world datasets exhibit similar robustness.

CERCS: P176 Artificial; Intelligence T111 Imaging, image processing.

Keywords: machine vision, depth estimation, video compression, occupancy prediction model

Contents

List of Figures

List of Tables

Abbreviations, terminology, constants

BEV - Bird's Eye View

CARLA - Car Learning to Act

JPEG - Joint Photography Expert Group

LiDAR - Light Detection and Ranging

MJPEG - Motion JPEG

RADAR - Radio Detection and Ranging

PNG - Portable Network Graphic

SDF - Signed Distance Function

Mbps - Megabits per second

FOV - Field of View

UDF - Unsigned Distance Function

1 Introduction

Machine perception systems are essential for any autonomous driving task. They enable the vehicle to sense and interpret its environment utilising a variety of cameras, Light Detection and Ranging (LiDAR), and Radio Detection and Ranging (RADAR) sensors to detect and track objects such as vehicles, pedestrians, and road infrastructure. This understanding is necessary for tasks such as obstacle avoidance, path planning, and decision-making. [1]

Many modern machine perception systems in autonomous driving research use deep learning methods to predict the location of nearby objects using only camera images. Camera-based occupancy prediction is attractive because cameras are cheaper and more widely available than sensors like LiDARs. Companies, such as Tesla [2], reportedly rely on vision-only systems, using neural networks to estimate the space around the car without extra sensors. [2]

Camera-based vision systems capture information about the environment through sequential images or videos. Uncompressed raw videos require high data rates, which can be mitigated by using video codecs that compress and decompress digital videos [3]. This technique is used in modern video streaming and storage, as encoded videos require less data to transmit and store [3]. It is reported that some lossy compression methods can reduce the video data up to 602.6 times compared to the uncompressed original size, depending on the complexity of the scene and the desired quality of the video [4]. Due to their effectiveness, video encoding can also help reduce the video data produced by camera feeds of autonomous driving systems.

1.1 Problem Formulation

Many video codecs implement lossy compression for encoding videos because of the compression efficiency [5]. Even though moderate video compression tries to maintain the visual fidelity for a human viewer [5], machines perceive visual media differently than humans. Therefore, machine learning algorithms performing tasks such as occupancy predictions on compressed camera frames could have a more adverse reaction to the loss in quality than expected.

1.2 Purpose of the Work

This thesis investigates how various video codecs affect the final predictions of a selected machine learning model for occupancy estimation. Existing research, using common image compression or video encoding algorithms, has analysed the compression effect on various object detection tasks [30, 32], but not specifically for occupancy prediction models that rely on camera frames for object distance estimation. The current work aims to bridge this gap and evaluate how compression-introduced quality degradation influences the reliability of predicted occupancy maps.

The experiments are performed on synthetic data gathered in the Car Learning to Act (CARLA) simulator [6], an open-source simulation software for autonomous driving research. Synthetic data eliminates the presence of sensor noise typically encountered when collecting data with real-world systems. This enables a focused analysis of the impact of compression noise on occupancy predictions, without the risk of other external noise sources influencing the results. Furthermore, a synthetic dataset is used instead of real-world datasets such as NuScenes [7] or Waymo OpenDrive [8], as these datasets only contain compressed JPEG images, without access to unprocessed camera sensor data. Although the dataset SemanticKITTI [9] includes raw 8-bit PNG images, these are still subject to color compression, as 8-bit color depth supports only 256 distinct colors, significantly fewer than the over 16 million colors possible with 24-bit color representation [10].

In summary, the thesis objective is to generate a simulation dataset for training an occupancy prediction model and then applying video compression to the dataset images to analyse the effect of different video codecs on occupancy prediction performance.

2 Background and Related Work

This chapter reviews foundational concepts in occupancy mapping, introduces widely used video compression standards, and summarizes recent findings on how compression influences machine learning performance in vision-centric tasks.

2.1 Occupancy Mapping and Prediction Models

Occupancy mapping is a task in autonomous driving systems that helps a vehicle or robot understand which parts of its surrounding environment are free to move through and which are blocked by obstacles. Dividing the space into a grid and recognizing obstacles in this grid representation provides a simplified way for machines to navigate safely and make decisions about movement. Occupancy grids are typically represented as two-dimensional or three-dimensional arrays, where each cell corresponds to a physical location in space. The cells hold a probability value indicating whether the space is occupied. [11]

2.1.1 Map Representations

Binary occupancy grid is a type of occupancy grid used for spatial occupancy in robotics and autonomous systems. It discretizes the environment into a 2D grid, where each cell represents a fixed area of space and each cell can only encode binary probability about the cell's occupancy status. This means that each cell can exist in one of two states: occupied or free. Conventionally, an occupied cell is assigned a value of 1, indicating the presence of an obstacle, while an unoccupied cell is represented by 0. The binary representation can produce unreliable results, as inconsistent sensor readings and the absence of confidence estimation can result in sequentially estimated grids looking dissimilar. This can be mitigated, however, by aligning the predictions across multiple readings. [11, 12]

Signed Distance Function (SDF) is a mathematical operation that calculates the shortest distance to the nearest boundary of an object [13]. In occupancy mapping, the distance is calculated from the occupied object surface, which provides a continuous approximation of the surrounding environment [14]. The advantage of distance transform based maps is that they essentially provide a better cost mapping for each cell in terms of distance from the object boundary [15] which many computer graphics and its related fields exploit for enhanced performance in tasks such as motion planning, mesh generation, and collision detection [16]. In contrast to binary occupancy grids, which merely indicate the presence or absence of obstacles, SDF maps offer more detail by assigning positive values to points outside obstacles, negative values to those inside, and zero to points on the boundary [16].

Unsigned Distance Function (UDF) is a mathematical operation that calculates the shortest distance for each cell to the closest surface or obstacle, without indicating whether the point is

inside or outside the object. Unlike the SDF, which encodes both the magnitude and the direction relative to the shape boundary, the UDF only captures the magnitude of the shortest distance. This makes it suitable for applications where the interior distinction is not required, or when the object shape is too ambiguous or noisy to define reliably. [17]

In this work, we define a Soft Occupancy function $S(x)$ as:

$$S(x) = \frac{1}{1 + UDF(x)}, \tag{2.1}$$

where

- $UDF(x)$ denotes the unsigned Euclidean distance from point x to the nearest obstacle boundary.

This inverse formulation of UDF assigns value 1 to obstacles and gradually closer values to 0 as the distance from the closest object increases. Unlike binary maps that sharply separate occupied and free space, the benefit of Soft Occupancy is the gradual gradient from occupied pixels. Furthermore, the presumed benefit over a UDF or SDF representation in the context of neural networks is its boundedness, which should work better for gradient descent. Soft Occupancy maps are used in this thesis because the selected occupancy prediction model relies on them for both training and prediction.

The comparison between BEV binary occupancy, SDF, UDF, and Soft Occupancy grid is visualized in Figure 2.1.

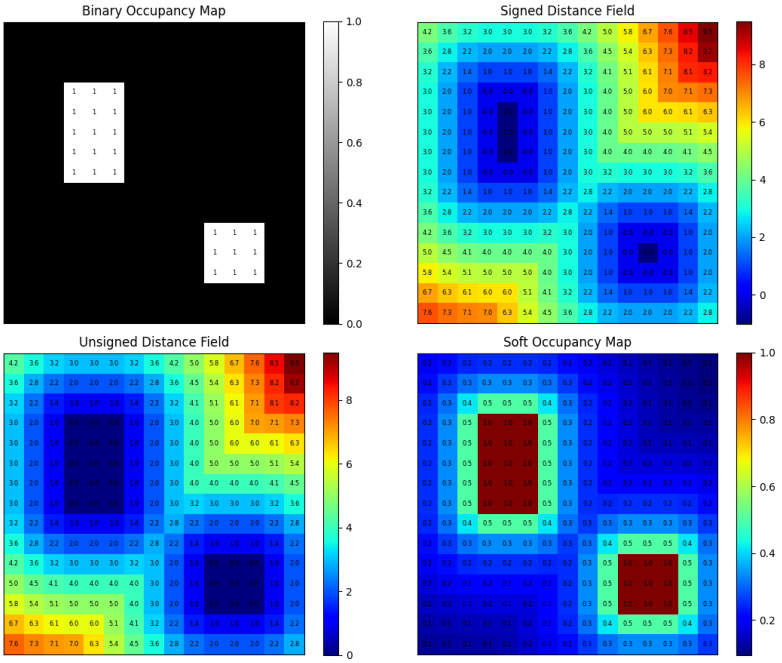


Figure 2.1: Examples of binary occupancy, Signed Distance Field, Unsigned Distance Field, and Soft Occupancy grid maps generated in Python using Matplotlib [29] library. Each representation shows each cell value using color coding, with appropriate numerical values shown in each cell.

2.1.2 Occupancy Prediction Models

Occupancy prediction models enable autonomous vehicles to comprehend and navigate their surroundings by forecasting the spatial occupancy and semantics of three-dimensional (3D) environments. Different models utilize various sensor modalities, including LiDAR and cameras. Multi-modal occupancy prediction systems fuse multiple sensor readings and information, which typically increases the model's reliability, as the system does not rely on a single type of sensor data, but consequently increases the computational complexity. Nevertheless, while LiDAR generally offers better distance estimations, it often comes with higher manufacturing and integration costs. In contrast, vision-centric or camera-based occupancy prediction models have gained prominence due to their cost-effectiveness, better real-time capabilities, and the dense semantic information that can be extracted from images. [2, 18]

Many multi-camera occupancy model architectures follow a similar training and prediction pipeline. First, two-dimensional (2D) image features are extracted from each camera. Then, these 2D features are projected into a target space, such as 3D space, Bird's Eye View (BEV), or three-plane view (TPV), through a process known as view transformation. In the target space, features from different cameras are spatially aligned in regions where multiple cameras have a field of view overlap. Finally, the fused features are decoded to predict, for example, semantic occupancy, enabling the classification of each region in the target space by labels such as road, vehicle, pedestrian, or building. [2, 18]

SimpleBEV [36] is a representative example of this architecture. This model projects 2D image features from all camera sensors into BEV space and aggregates them. The model learns based on LiDAR or other 3D sensing modalities that produce adequate geometry estimation. In this thesis, I use a modified version of the SimpleBEV model, which has only been applied in internal experiments in Tartu Observatory and has not been officially released. While based on the original SimpleBEV framework, the version employed here includes unpublished adjustments specific to our task. As this thesis focuses not on the model itself, only the necessary details relevant to the compression study are described. The modifications are explained in Chapter 3.3.

2.2 Video Compression

Video compression falls into two main categories: lossless and lossy. Lossless compression restructures video data by eliminating redundancy and encoding it more compactly, enabling the original content to be identically reconstructed later. Lossy compression achieves higher compression rates by combining lossless techniques with additional processes that discard certain details during encoding. While these lost details cannot be recovered, lossy methods are designed to ensure that moderate levels of compression do not noticeably impact the perceived visual quality for human viewers, despite the objective reduction in video quality. [5]

2.2.1 Spatial and Temporal Compression

Spatial compression, also known as intra-frame compression, aims to reduce redundancy contained within individual video frames by exploiting the similarity between neighboring pixels. Individual video image frames typically exhibit local continuity, where adjacent pixels tend to have similar color and intensity values. Compression algorithms leverage this phenomenon to reduce data without perceptually degrading image quality. [20]

Common operations in spatial compression are quantization and transform coding, which reduce the precision of transformed image data. By transforming the pixel intensity values into the frequency domain through transform coding methods such as Discrete Cosine Transform and downsampling visibly less noticeable frequencies through discretization, the quantization discards less perceptually significant information [21]. While these steps are essential for achieving high compression ratios, they also introduce a quality loss, which can not be restored during video reconstruction. Algorithms performing intra-frame compression can directly influence how quantization and transform coding are applied to balance compression efficiency and visual fidelity. [20]

Another important step in spatial compression is transforming image data into a more compression-friendly color space format. One such transformation involves converting Red, Blue, and Green (RGB) color model values into the YUV color space. In this representation, the Y component represents brightness, while the U and V components represent color difference. This separation aligns with the characteristics of human visual perception, which is more sensitive to changes in brightness rather than color. It is important to note that the more specific term for the color space for digital media is YCrCb, which approximates the original YUV implementation [33]. In this work, the term "YUV" is also used to refer to the YCrCb color space, as one of the software tools used in this thesis, FFmpeg [22], employs the more generic term. The term "YUV" will be used throughout to maintain consistency and avoid confusion. [20]

In the 4:4:4 YUV format, all three channels are sampled at every pixel, preserving full color information. This format offers the highest fidelity but results in the largest file sizes. By contrast, the 4:2:0 format reduces data requirements by downsampling the U and V components. Specifically, for every two-by-two block of pixels, all four pixels share a single U value and a single V value, while each pixel retains its unique Y value. This leads to better compression gains while retaining image quality that is typically indistinguishable from formats under normal circumstances. [20]

Temporal compression, also known as inter-frame compression, targets redundancy by reducing the amount of data stored across successive video frames. Unlike spatial compression, which operates within a single frame, temporal compression exploits the fact that much of the visual content in a video remains static or undergoes predictable motion over time. [20]

2.2.2 Video Codec Standards

Video Codec is a compression standard that defines how to encode video information into an efficient representation, and how to decode that representation back into a form that restores or approximates the original video. Different standards utilize different spatial and temporal compression techniques to reduce data whilst maintaining quality. [5]

Motion JPEG (MJPEG) compresses video by encoding each frame as an individual JPEG image. This approach results in minimal latency, which is advantageous for real-time applications requiring immediate frame access, such as live monitoring and control systems. However, the absence of temporal compression leads to larger file sizes and higher bandwidth consumption than more advanced codecs. Furthermore, MJPEG does not allow fixed bitrate control, making it less desirable for applications with strict bandwidth constraints. Despite these drawbacks, MJPEG's simplicity and low latency make it a viable option for scenarios where immediate frame availability outweighs storage efficiency. [34]

H.264, also known as Advanced Video Coding (AVC), employs both spatial and temporal compression techniques, such as motion estimation and compensation, to reduce redundancy between frames. This results in smaller file sizes and efficient bandwidth usage, making H.264 suitable for various applications, including streaming and storage. For real-time systems, H.264 balances compression efficiency and latency, especially when configured with low-latency settings and hardware acceleration. [34]

H.265, or High Efficiency Video Coding (HEVC), is the successor to H.264 and offers more efficient coding, resulting in a better compression ratio at better levels of quality. It achieves this through advanced techniques, such as improved motion vector prediction and larger coding block sizes, enabling higher quality video at lower bitrates. However, the increased computational complexity of H.265 may introduce higher latency, making it less suitable for real-time applications without specialized hardware support. Therefore, while H.265 offers superior compression, its applicability in real-time systems depends on the availability of resources to mitigate latency concerns [34].

These codecs were selected based on their relevance in prior research. Studies have examined the impact of JPEG and H.264 compression on object detection, revealing minimal performance degradation under moderate compression levels [30, 32]. H.265, the successor to H.264, offers improved compression efficiency and represents the next logical step in video encoding standards for machine learning and autonomous driving applications.

2.2.3 Video Encoding and Decoding using FFmpeg

FFmpeg is an open-source multimedia framework for video and audio recording, processing, and streaming [22]. Among other useful features, the software can encode and decode videos and supports various compression standards. In particular, FFmpeg supports MJPEG, H.264, and H.265 video codecs [23]. Videos can be encoded from a set of source images or another video.

FFmpeg allows significant control over the encoding process and compression rate. This thesis aims to control the image quality degradation and encoding-decoding speed. The encoder for MJPEG videos allows control of the quality factor parameter ”q”, which is used to specify the compression rate on a scale of 1-31, where smallest value describes highest level of quality with least aggressive compression and highest value specifies lowest quality with most aggressive video data compression [24]. The encoders for both H.264 and H.265 encodings utilize the Constant Rate Factor (CRF) parameter to control quality and compression ratio. Similarly to the MJPEG quality parameter, the CRF parameter controls the balance between video quality and compression, with lower values indicating higher quality and less compression, and higher values resulting in greater compression at the cost of visual fidelity. The CRF scale in FFmpeg ranges from 0 to 51, where 0 is lossless compression and 51 is the most aggressive compression producing the worst quality. The default value in FFmpeg is 23, offering a good trade-off between quality and file size [31]. Furthermore, FFmpeg allows the selection of different YUV color subsampling schemes through a parameter called ”pix_fmt” or pixel format. The color subsampling scheme used in this thesis is 4:2:0 color subsampling, which can be enforced by setting the pixel format parameter to the value ”yuvj420p”. If full color resolution is required, the full 4:4:4 color representation can be called using the value ”yuvj420p”. Finally, for H.264 and H.265, it is important to set encoding latency constraints for real-time applications. Without the latency constraints, the encoder buffers camera frames to apply better temporal compression

at the expense of increasing encoding delay. Minimizing the delay is necessary for real-time robotic systems to provide the perception system with frequent frames to make timely occupancy predictions. FFmpeg allows ”preset” and ”tune” parameters to adjust the encoding time and buffering. The ”preset” parameter ranges from scale ”very_slow” to ”ultra_fast” to specify the encoding speed with respect to the rate of compression. The ”tune” parameter has many different modes like ”film”, ”stillimage”, or ”zerolatency”. It allows for fine-tuning the encoding process for the specific use case. This thesis uses the ”ultra_fast” preset with ”zerolatency” tuning to ensure minimal possible latency. These parameters allow control of camera sensor video quality and compression artefacts.

2.3 Related Works on Compression and Object Detection

Existing research has studied the influence of JPEG lossy image compression on deep learning models for object detection. Encoders for JPEG compression allow defining the level of compression by specifying the encoder quality factor parameter on a scale of 0 to 100, where value 0 represents the most aggressive compression and value 100 represents the highest image quality with the least compression. A study has shown that object detectors maintain stable performance when JPEG quality levels are above 25–30, but experience a sharp degradation when quality drops below this threshold. Specifically, models show relatively minor losses between JPEG quality settings of 100 and 30, but object detection accuracy declines significantly below a quality setting of 20, especially for small or fine-structured objects. Furthermore, the degradation occurs abruptly once the compression becomes too aggressive. [30]

Similarly to image compression, a study on video compression using the H.264 standard demonstrated that object detection systems can tolerate considerable compression before experiencing performance loss. The study encodes uncompressed videos to the H.264 format using the x264 encoder, which allows influencing video compression rate through the CRF parameter. The research investigated detection model performance on H.264 videos encoded with CRF values 22, 32, 37, 42, and 47. The detection model showed little to no performance reduction with videos encoded using CRF values 22, 32, and 37. However, detection performance began to degrade substantially at higher compression levels corresponding to CRF values of 42 and 47, especially for scenes with challenging illumination or fast-moving objects. [32]

The findings from prior studies on object detection under image and video compression are directly relevant to this work, as they demonstrate how compression affects the performance of deep learning models depending on compression strength and scene characteristics. Although occupancy prediction differs from object detection, both tasks rely on extracting meaningful spatial features from visual data. These results provide a useful starting point for designing compression tests to analyze the compression effect on occupancy models.

3 Methodology

Video compression effects on multi-camera deep learning occupancy prediction model experiments are performed in 5 steps.

1. Simulation execution. The CARLA simulation is set up, and a simulation scenario is created. A vehicle with a sensor configuration is spawned, and sensor information is gathered throughout the driving scenario.

2. Data processing. Simulation data is processed, and ground truth occupancy grids are generated for model training.

3. Video compression. Raw camera images from the simulation are compressed using different video encodings to varying levels of quantization and color subsampling. The compressed images are saved and used for model training.

4. Model training. The occupancy model is trained using the ground truth generated in the data processing step and compressed frames from the video compression step. The training produces evaluation metrics for training and validation sets that need to be evaluated.

5. Model evaluation. Evaluation metrics are visualised and analysed for every individual video compression configuration.

The following sections explain each step in more detail.

3.1 Simulation and Data Collection Setup

CARLA is an open-source simulator for autonomous driving research for developing, training, and validating autonomous urban driving systems. It provides a suite of sensors, including RGB cameras, depth sensors, LiDAR, ground truth annotations like bounding boxes, and semantic segmentation. CARLA remains a reputable open-source simulator for research, with good documentation and community support, which supports the development of flexible driving scenarios and utilizes various sensor configurations. [35]

3.1.1 Ego Vehicle and Sensor Configuration

Simulated ego vehicle in the CARLA simulator contains a sensor suite that records sensor readings throughout the driving scenario. The current experiment bases its 6-camera and LiDAR sensor configuration on the NuScenes autonomous driving dataset vehicle. The specific values in sensor position and camera specification derived from the actual NuScenes dataset are described in Appendix A. The visualization of the NuScenes sensor configuration is shown in Figure 3.1.

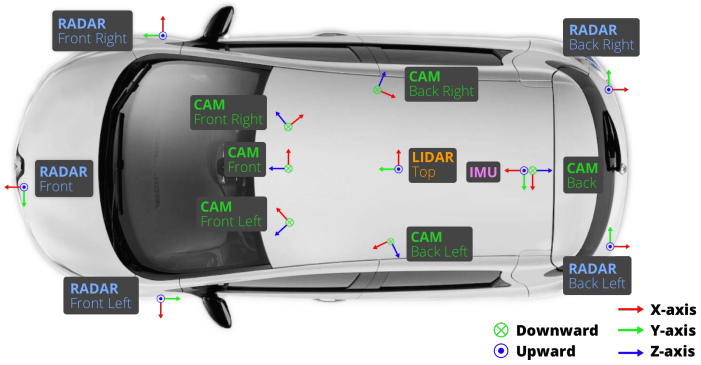


Figure 3.1: Illustration of NuScenes vehicle sensors and their positions. [7]

One significant change to the simulation vehicle setup differs from its real-world counterpart. Every camera is virtually equipped with a depth image sensor and a semantic image sensor that record depth and semantic values for the corresponding color (RGB) image (see Figure 3.2). This is used in the data processing portion of the experiment setup, where the occupancy grid of obstacles is generated from these additional views instead of utilizing LiDAR point clouds.

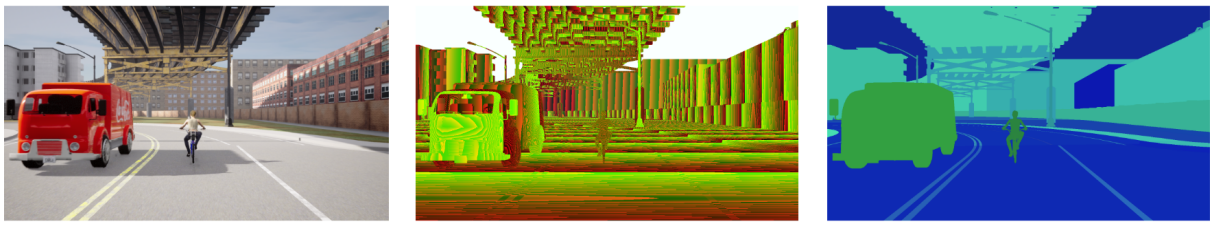


Figure 3.2: Simulation output from the front camera. The left-most frame shows the color image, the middle displays the depth image, and the right-most frame presents the semantic segmentation of the simulated scene.

3.1.2 Data Collection Process

CARLA simulator allows the creation of driving scenarios and collects sensor readings from the runtime. The simulator is run in synchronous mode, allowing simulation time control through "ticking". Each time the simulator receives the tick signal, a predefined period will elapse, and the simulator simulates the movement of each dynamic object. The current simulation is set up so that each tick elapses 0.1 seconds, and the changes, like car and pedestrian movement, are simulated. After each tick, the sensor readings from the simulation are retrieved and stored. The whole simulation process can be described in the following steps:

1. The CARLA simulator is started
2. The simulation environment is loaded. In this experiment, we load an environment called "Town 3", as it is one of the largest environments shipped with the simulator.
3. The simulation is set up by spawning 20 vehicles and pedestrians. Each vehicle and pedestrian is attached to an AI control system called TrafficManager [25], which controls the movement for these simulated actors.

4. The ego vehicle is spawned. This vehicle contains all of the necessary sensors for generating the synthetic dataset. The vehicle has a preplanned path (see Appendix B) that was created before the simulation. It drives to each path location using an autonomous control system called BasicAgent [37], which controls the movement and behaviour of the ego vehicle and allows it to navigate to each location using the shortest path possible with legal traffic maneuvers.

5. The simulation is started. Simulation remains static until the simulator is given a tick signal. The image information from cameras and the point cloud information from LiDAR are stored for each tick signal. In addition, a 4x4 transformation matrix is extracted for every sensor, which contains the location and orientation information for each sensor in reference to the origin point of the map. This allows tracking the movement of each sensor throughout the driving scenario.

6. The simulation is run until the driving scenario is complete. The simulator is ticked, and data is gathered until the ego vehicle has reached the final location of the predefined path.

The resulting raw simulation data should contain the necessary camera and LiDAR sensor readings for training the occupancy prediction system. However, some data processing is required to extract the required information to run the machine learning pipeline.

3.1.3 Data Processing

The simulation produces raw sensor outputs necessary for training the vision-based occupancy prediction model. However, this raw data must first be processed to generate the ground truth smooth occupancy maps to supervise the deep learning model.

The ground truth smooth occupancy maps are generated from depth and semantic images of the simulated camera output. These camera views, in combination with camera intrinsic values, are used to project the semantic values into 3D space as a point cloud, as shown in Figure 3.3. The colors in the semantic image represent each object's semantic label and instance identification. Therefore, every object category and unique identifier is represented in the color information.

Since the configuration uses six cameras, the projected semantic point clouds are then merged together, and the point cloud is transformed around the recorded LiDAR position, as the LiDAR sensor serves as a center point and provides a forward direction for the sensor configuration. Figure 3.4 visualizes the full semantic point cloud.

The full semantic point cloud enables the creation of the necessary Soft Occupancy map representation for the machine learning model. As each semantic color in the point cloud contains an object category, the distinction between ground plane elements and obstacles can be made. Filtering the obstacle points using the encoded color information can effectively produce a point cloud that captures all points considered obstacles. The obstacle points are then processed further to eliminate points that are either less than 0.3 meters or more than 1.5 meters above ground, as the points outside this region should not interfere with the vehicle's path and are therefore not considered obstacles. After processing the camera information into an obstacle point cloud, the binary occupancy grid map is generated from it. The rasterization from 3D points to a 2D grid map is achieved by discarding the height (Z) coordinate for each point, and then the resulting X and Y coordinates are discretized to the nearest 0.5 m cell. The resulting grid size is set to 104 by 104 pixels. The resulting grid map describes an area of 52 by 52 meters. The final Soft

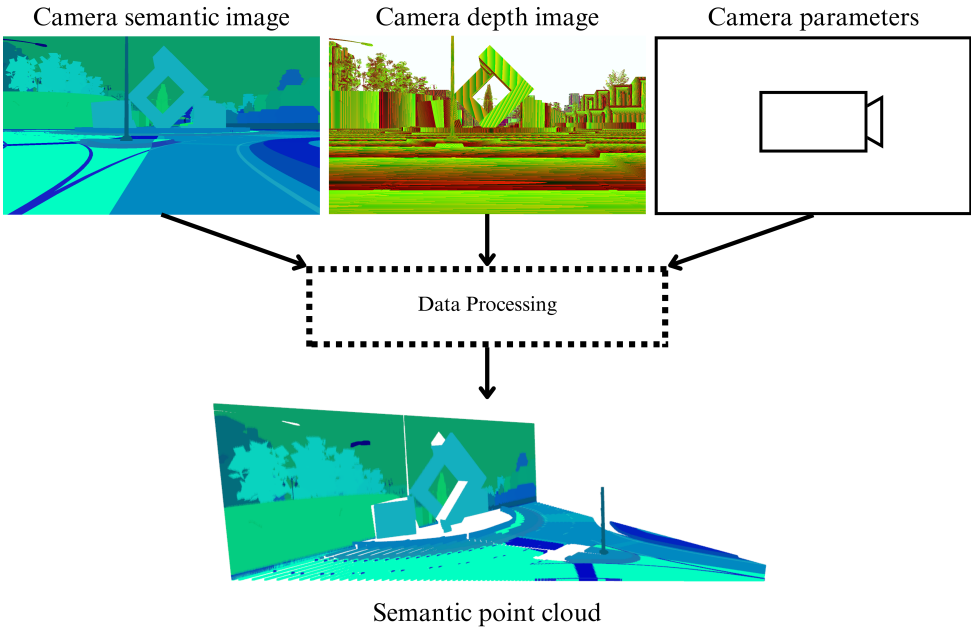


Figure 3.3: Single camera sensor readings projected into 3D space

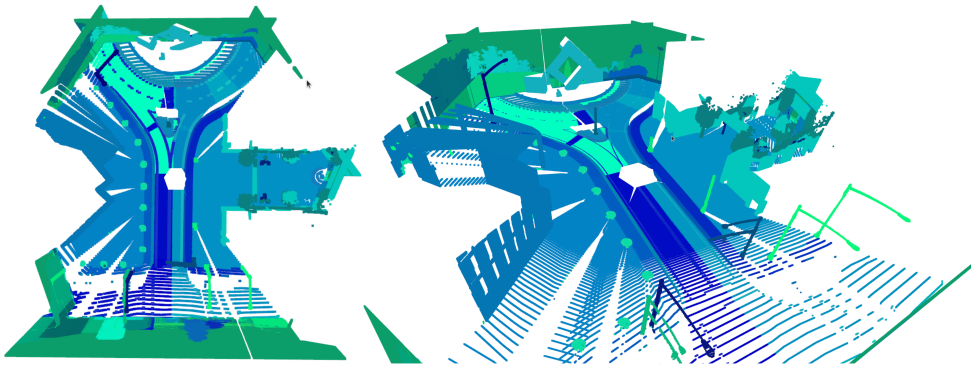


Figure 3.4: Merged point clouds from all cameras

Occupancy grid map, necessary for the occupancy model training, is generated by applying the Soft Occupancy function to the previously generated binary occupancy grid map. Figure 3.5 contains the visualization of this process.

A binary grid map representation called a visibility mask is another necessary map representation for the Tartu Observatory model. Visibility masks identify the BEV grid portions visible to each camera at a given timestamp. The mask indicates which areas of the environment should contribute to the training signal from each camera frame, enabling the network to differentiate between observed and unobserved regions. Similarly to Soft Occupancy maps, the visibility mask is also generated from the semantic point cloud generated by the camera projections. Each point in the camera's viewing space is ray-traced, and the resulting rays are sampled into a visibility point cloud representation. The resulting point cloud is then filtered by removing points higher than 1.5 meters above ground level to maintain the height restriction imposed on the obstacle point cloud. The visibility point cloud is then rasterized to the 2D grid map identically to the binary occupancy map described previously. Figure 3.5 contains the visualization for this process.

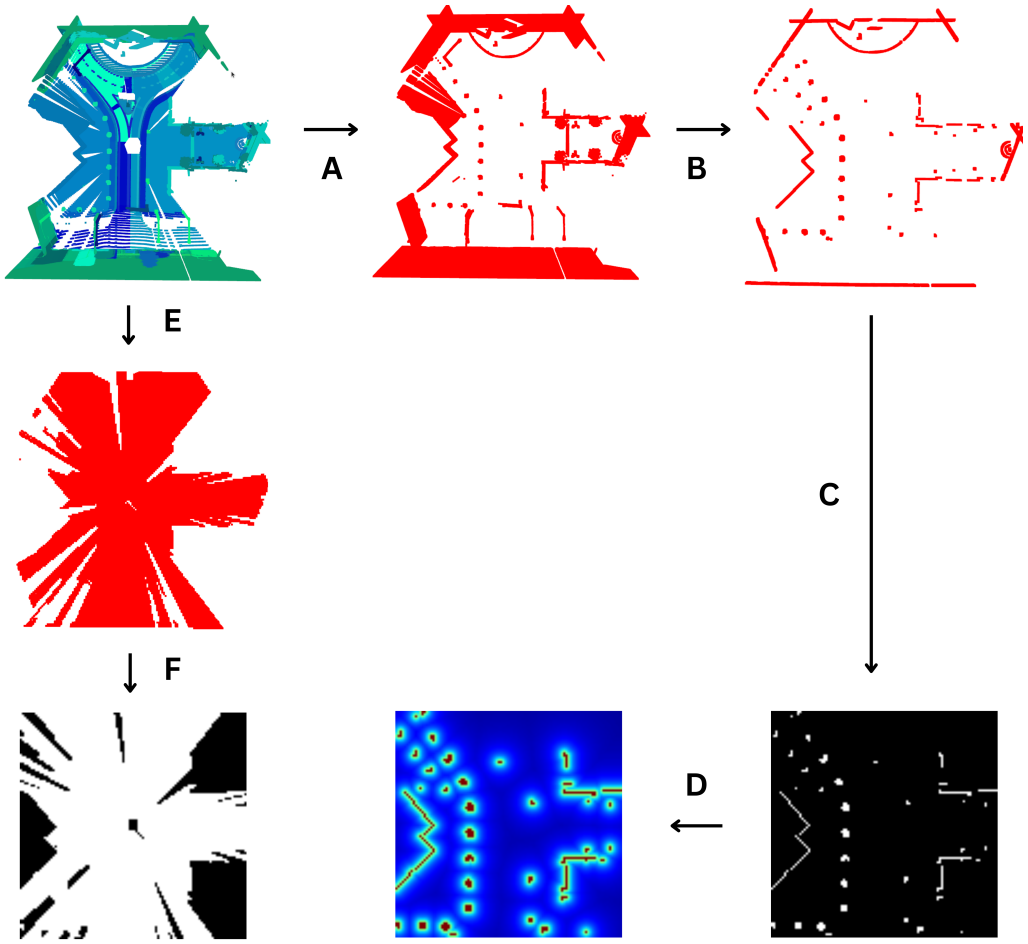


Figure 3.5: Illustration of soft occupancy map and visibility binary map generation process. Semantic point cloud is filtered for obstacle points (A). The Resulting obstacle point cloud is filtered so that points that are either too high or too close to the ground are excluded (B). The cropped point cloud is rasterized to a binary occupancy grid (C). Then the soft occupancy function is applied to the binary representation to produce a soft occupancy map (D). The visibility mask is generated by ray-tracing each camera and generating a point cloud by sampling points on the rays (E). The sampled point cloud is rasterized to a binary visibility grid mask (F).

This data processing pipeline transforms the simulation-generated raw sensor data into structured training targets. These outputs are necessary for the deep learning model to learn spatial reasoning and occupancy prediction.

3.2 Video Compression Pipeline

Creating a video compression pipeline is necessary to analyze lossy compression on a vision-based occupancy prediction model. The raw camera images collected from the CARLA simulation environment are initially stored as uncompressed RGB frames. The pipeline aims to encode these raw image sequences into videos using various video codecs to simulate compression scenarios that might occur in real-world systems due to bandwidth or storage constraints.

The goal of this pipeline was to produce compressed versions of the synthetic dataset. By compressing synthetic camera images into video data using video codecs that employ lossy compression, decoding them back into image frames, the resulting decoded frames contain all the compression artefacts that occur due to the lossy nature of the video encoding process. This setup allows a comparison between models trained using different compression inputs.

3.2.1 Compression Tools and Settings

The FFmpeg multimedia framework was selected for video compression because it supports the necessary codecs and the control over encoding parameters. Three video codecs were selected for experimentation: MJPEG, H.264, and H.265. Each codec represents a different trade-off between compression efficiency and latency. MJPEG performs intra-frame compression only, treating each frame as an independent JPEG image. This results in minimal latency but lower compression efficiency. On the other hand, H.264 and H.265 employ both spatial and temporal compression techniques, achieving much higher compression ratios but at the cost of increased computational complexity.

For the MJPEG codec, the videos were created by manipulating the video encoder quality scale ”q” parameter to adjust the video’s compression rate and visual quality. The selected levels for compression were 1, 16, and 31 as they represent the full quality scale provided by the encoder. The quality level 1 offers the best quality with the least compression, and level 31 provides the most aggressive compression with degraded video quality.

For H.264 and H.265, the videos were encoded by adjusting the video encoder CRF parameter. The selected compression levels were adjusted by compressing the frames using CRF levels 2, 23, and 51. Similarly to the MJPEG quality scale, the CRF parameter values range from least compression to most compression, where 2 is almost lossless quality, 23 is the FFmpeg default CRF value, and 51 is the most compressed output with the least quality. Additionally, the parameters ”tune” and ”preset” were adjusted to settings ”zerolatency” and ”ultrafast”, respectively. These parameters limit the internal buffering during encoding to ensure real-time video processing settings.

The color subsampling effect was also analysed with the selected video codecs. The MJPEG and H.264 encoded videos were created using both 4:2:0 color subsampling (YUV420) and no color subsampling 4:4:4 (YUV444). This allows for analysis of how this color downsampling for images influences the occupancy prediction model. The H.265 was only encoded using

4:2:0 subsampling due to time constraints.

All 15 variations of video encoding settings used in the video compression pipeline are described in Table 3.1. The encoding and decoding operations were executed on an in-house machine learning server. The server is equipped with an Intel Xeon Gold 6426Y processor. During both of the processes, FFmpeg produced benchmarking results regarding encoding and decoding speed. The results were recorded and analysed to gain a basic understanding of the overall performance of the video codecs in comparison to each other. However, since the video compression pipeline was run only once, the results are interpreted with caution. Single-run benchmarks are susceptible to external processes influencing the results and thus skewing the typical performance metrics.

Encoding Settings				
Encoder	Video Codec	Video Quality Parameter	Video Color Subsampling	Video Encoding Additional Parameters
mjpeg	mjpeg	q=1	yuvj420p	(none)
mjpeg	mjpeg	q=1	yuvj444p	(none)
mjpeg	mjpeg	q=16	yuvj420p	(none)
mjpeg	mjpeg	q=16	yuvj444p	(none)
mjpeg	mjpeg	q=31	yuvj420p	(none)
mjpeg	mjpeg	q=31	yuvj444p	(none)
libx264	h.264	crf=2	yuvj420p	-tune zerolatency -preset ultrafast
libx264	h.264	crf=2	yuvj444p	-tune zerolatency -preset ultrafast
libx264	h.264	crf=23	yuvj420p	-tune zerolatency -preset ultrafast
libx264	h.264	crf=23	yuvj444p	-tune zerolatency -preset ultrafast
libx264	h.264	crf=51	yuvj420p	-tune zerolatency -preset ultrafast
libx264	h.264	crf=51	yuvj444p	-tune zerolatency -preset ultrafast
libx265	h.265	crf=2	yuvj420p	-tune zerolatency -preset ultrafast
libx265	h.265	crf=23	yuvj420p	-tune zerolatency -preset ultrafast
libx265	h.265	crf=51	yuvj420p	-tune zerolatency -preset ultrafast

Table 3.1: Video Encoding Settings

3.3 Training and Evaluation Strategy

The occupancy prediction model, which is used for the experiments, is trained to predict Soft Occupancy and visibility grids using 6 RGB camera views. The selected model uses a composite loss function for training and validation. It combines four components: Mean Squared Error (MSE) loss, Gradient loss, Hessian loss, and Binary Cross-Entropy (BCE) loss for visibility. These losses are aggregated into a single validation loss to facilitate the joint optimization of geometric accuracy and visibility estimation. The general formula for this compound loss function can be described as:

$$L(\theta) = \lambda_{\text{MSE}}\, L_{\text{MSE}}(y, \hat{y}) + \lambda_{\text{grad}}\, L_{\text{grad}}(y, \hat{y}) + \lambda_{\text{hess}}\, L_{\text{hess}}(y, \hat{y}), + \lambda_{\text{BCE}}\, L_{\text{BCE}}(y, \hat{y}),$$

where

- L_{MSE} measures per-pixel squared error for global fidelity;
- L_{grad} penalizes differences in first-order derivatives to preserve edges and fine structures;
- L_{hess} penalizes differences in second-order derivatives (the Hessian) to maintain smoothness and curvature;
- L_{BCE} penalizes the misclassification of binary visibility mask labels.

The MSE loss measures the average squared difference between the predicted and ground truth Soft Occupancy maps, guiding the network to produce spatial predictions that are accurate across the entire predicted Soft Occupancy map. To further guide the model toward learning fine-grained geometric structure, the model uses a gradient loss component that penalizes discrepancies in the first-order spatial derivatives. This helps enforce sharper object boundaries and better local consistency. Complementing this, the Hessian loss captures differences in second-order derivatives, promoting the learning of smooth and coherent surface transitions,

particularly in regions of complex geometry. The binary visibility map is guided by the BCE loss. This loss treats each grid cell as a binary classification task, guiding the model to distinguish between visible and occluded regions based on the camera viewpoint and scene layout.

Together, these four loss components are aggregated into a single loss value, and they provide a training and validation signal for the occupancy model. While the MSE ensures global alignment with the ground truth, the Gradient and Hessian losses enhance structural detail and continuity, and the visibility BCE loss contributes to semantic understanding of occlusion. This multi-term loss formulation is used for evaluating the degradation in spatial prediction performance resulting from video compression artefacts.

The training strategy is designed to directly measure the effect of different video compression codecs and parameters on occupancy prediction performance. Separate training runs are conducted using frames extracted from videos encoded with MJPEG, H.264, and H.265, each at multiple quality or compression levels.

The evaluation strategy focuses on a comparative analysis of the final performance across datasets encoded with different video compression settings. Specifically, we track and compare training and validation losses throughout the training process for each codec and compression level. In addition to quantitative loss analysis, the work provides a qualitative comparison between trained model outputs to better visualize the differences and similarities between each model's characteristics.

It is important to emphasize that the development and implementation of this machine learning model is not part of the thesis scope. The experimental model is developed by machine learning researchers of Tartu Observatory.

4 Results and Analysis

This chapter presents the outcomes obtained from evaluating the effect of video compression on a vision-based occupancy prediction model. The results are structured to reflect the various stages of the completed work, beginning with the generation of a synthetic dataset using the CARLA simulator, followed by the creation of multiple compressed versions of the dataset using various video encoding configurations. Afterwards, the performance of the occupancy model is analyzed across these dataset variants in terms of training and validation losses, as well as qualitative prediction accuracy. Particular emphasis is placed on identifying how compression parameters—such as codec type, quality level, and chroma subsampling—impact the model's ability to accurately predict occupancy and visibility maps. The findings serve to quantify the trade-offs between compression efficiency and predictive reliability.

4.1 Synthetic Dataset

This section outlines the creation of a synthetic dataset used to evaluate the impact of video compression on occupancy prediction. Using the CARLA simulator, a multi-camera setup on a simulated ego vehicle collected RGB, depth, and semantic images during a predefined driving scenario. These data were processed into ground truth Soft Occupancy maps and binary visibility grid maps. The resulting dataset was then encoded using various compression settings to produce multiple versions for model training and evaluation.

Created source code for running the CARLA simulation scenario for synthetic sensor data collection and data post-processing was uploaded to a GitHub code repository. Specific resource links and instructions are provided in Appendix A. The automated script for encoding all camera images into a video, with different video codec configurations, and then decoding the images back to individual camera frames is provided in Appendix D.

4.1.1 Raw Simulation Data

The CARLA simulation scenario produced the necessary camera and LiDAR data. In total, the simulation generated around 22 minutes of driving data with 13138 data points. The readings for each sensor were stored, with each reading containing the simulation timestamp as a filename. Stored data included 24-bit RGB color, depth, and semantic images retrieved from all six cameras, in addition to LiDAR point clouds and global pose information, which was represented as a 4x4 transformation matrix, for every sensor. An example of produced views for a single camera is visualized in Figure 3.2. An example output of all cameras can be seen in Appendix C.

4.1.2 Simulation Dataset

Raw simulation data was processed into a simulation dataset. This process required creating Soft Occupancy maps, along with visibility binary grid maps from simulation sensor readings at each timestamp.

After generating the necessary ground truth data, the dataset is ready to be exported for model training. The extraction process exports 13138 images from all cameras, and their corresponding Soft Occupancy and visibility maps. The dataset also includes auxiliary map representations like FOV and BEV masks, but in terms of model training and evaluation, this content is ignored and not used. The resulting dataset can be seen in Figure 4.1.

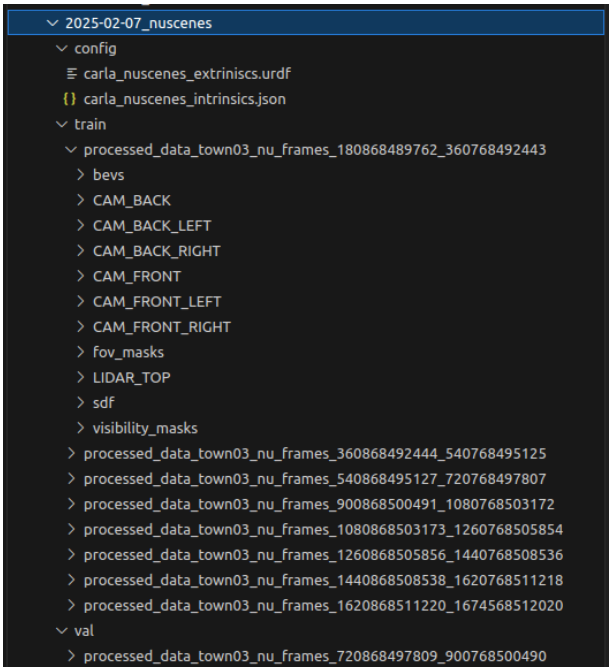


Figure 4.1: Dataset directory structure

4.1.3 Compressed Training Dataset Versions

The initial exported dataset contained uncompressed camera images. These images were compressed using 15 different video encoding configurations, and the decoded images were saved separately for the soft occupancy model training. The comparison of compression introduced artefacts can be seen in Figure 4.2 where a small region of a single frame is compared between different compression settings. The full images are available in Appendix D in Table D.1, Table D.2, and Table D.3.

Many of the compressed frames are visually indistinguishable from the original uncompressed image. The exceptions are the H.264 and H.265 encoded images at CRF level 51 that show noticeable blocking, color banding, color smearing, and discoloration when compared to less compressed versions of images. With MJPEG compression, there is a more subtle blurring with moderate and high compression images, but other quality concerns are very subtle. The visual comparison should serve as a representation of different levels of compression.

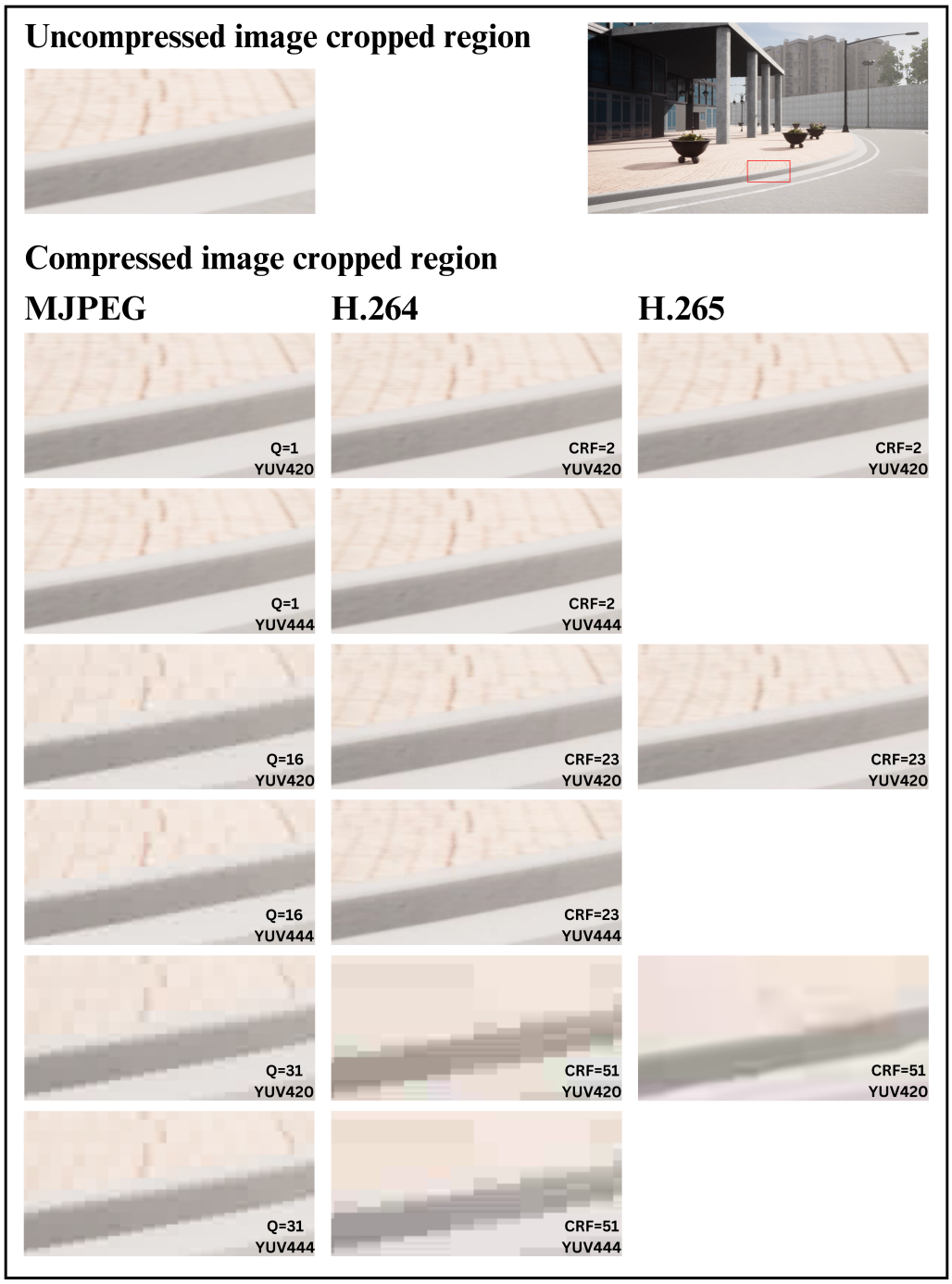


Figure 4.2: Comparison of video compression effect on image quality. Figure contains a cropped region of interest for uncompressed camera frame and camera frames that underwent compression using MJPEG, H.264, and H.265 codecs with different compression settings.

The comparison between different compression rates is visualized in Table 4.1.3. The table shows that the H.264 and H.265 encoders, which utilize the CRF parameter, allow for a wider range of compression. While encoders like libx264 produced H.264 encoded datasets, where average camera bitrate ranged from 169.26 Mbps to 0.15 Mbps, the MJPEG encoder managed to produce camera streams where bitrate only ranged from 19.53 Mbps to 2.54 Mbps. This consequently explains why in Figure 4.2 the images from videos with encoding CRF value of 51 look more distorted than any MJPEG images, as the video these images were decoded from

has noticeably more information in terms of bitrate. The moderate compression range, where bitrate stays between 2.5 and 7.5 Mbps, should offer a good comparison range across all selected encoders and provide a good reference point to assess the video compression effect on occupancy model performance.

Video Encoding	**Average Bitrate (Mbps)**
H.264 CRF=2 YUV444	169.26
H.264 CRF=2 YUV420	157.28
H.265 CRF=2 YUV420	44.08
MJPEG Q=1 YUV444	19.53
MJPEG Q=1 YUV420	15.24
H.264 CRF=23 YUV444	6.18
H.264 CRF=23 YUV420	5.37
MJPEG Q=16 YUV444	4.92
MJPEG Q=31 YUV444	3.89
MJPEG Q=16 YUV420	3.51
H.265 CRF=23 YUV420	2.86
MJPEG Q=31 YUV420	2.54
H.264 CRF=51 YUV444	0.19
H.264 CRF=51 YUV420	0.15
H.265 CRF=51 YUV420	0.05

Table 4.1: Average bitrate (Mbps) for different encoding settings sorted by bitrate in descending order.

4.2 Occupancy Model Training

The occupancy model was trained on 15 variations of the simulated dataset, where each dataset variant contained camera images that underwent different configurations of video compression. The training targets, like Soft Occupancy maps and visibility masks, remained identical for all model training iterations. Throughout the training process, training and validation losses were recorded as their metrics for the basis of the performance analysis. The best models for each training run were stored separately and were later used to output predicted Soft Occupancy and visibility masks for the validation set for qualitative analysis.

4.2.1 Training and Validation Loss Performance

The training loss curves for each dataset variant are presented in Figure 4.3. These curves show that all models followed a similar learning trajectory, with comparable patterns in spikes and troughs. This consistency suggests that, despite varying compression artefacts, the model's learning dynamics remained largely unaffected. A key difference across training runs was the number of steps required for convergence. Due to early stopping criteria, where training was halted if validation loss did not improve over five consecutive epochs, some models terminated at an earlier training step than others.

The best validation loss for each training run is illustrated in Figure 4.4. The compound loss metric is plotted against the average bitrate of the corresponding compressed video streams for

each training run. The results demonstrate that most models maintained relatively similar validation loss values until bitrate reduction through compression became too severe. In particular, models trained on H.264 and H.265 videos with a CRF value set to 51 exhibited a noticeably higher validation loss between 2.01 and 2.21. In contrast, models trained on datasets with average bitrates in the 2.5-7.5 Mbps range achieved validation losses between 1.10 and 1.35, which is near the uncompressed baseline validation loss of 1.14. One MJPEG-encoded dataset produced even lower validation loss than the model trained on the uncompressed dataset. The graph also demonstrates that color subsampling (YUV420) produced better validation losses in 4 out of 6 cases at lower bitrates, where a comparable model with no color subsampling (YUV444) existed.

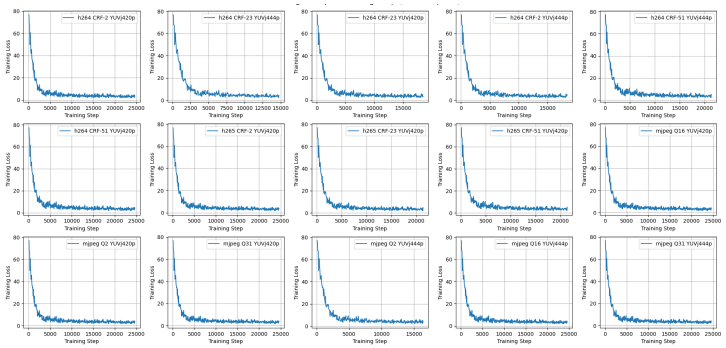


Figure 4.3: Training loss for datasets with different video encoding compression applied to camera frames.

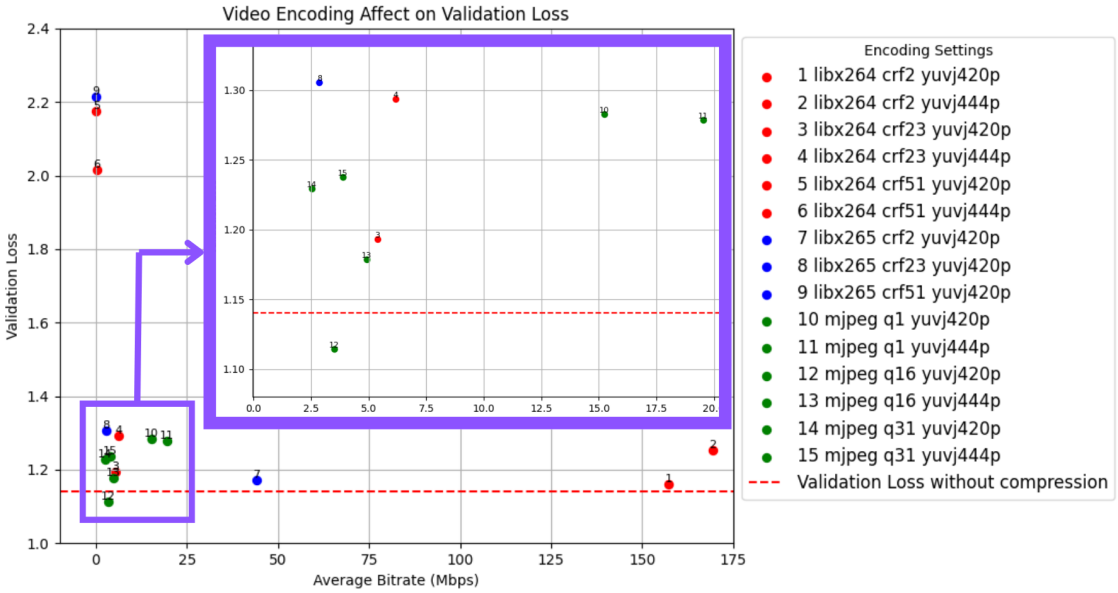


Figure 4.4: Validation loss for datasets with different video encoding compression applied to camera frames.

4.2.2 Visualization of Prediction Maps

The trained occupancy models that produced the lowest validation loss were used to visualize the validation set predictions to gain a better spatial understanding of the loss data. Figure 4.5 highlights the best and worst model prediction performance by comparing their 3-second sequence of predicted Soft Occupancy maps to the expected ground truth Soft Occupancy maps. Interestingly, all the models, even the ones trained on visually degraded images, produced outputs that had structures resembling the expected outcome. The main visual distinction between different model predictions was the temporal stability of the output. Static structures like walls exhibited a distinctive warping effect, where the boundary of the object seemingly changed over multiple different predictions. Still, the predicted maps maintained overall visual consistency with the expected ground truth result.

The visualization of an example prediction for all 15 models can be seen in Appendix E in Table E.1, Table E.2, and Table E.3. Every model output is not individually analysed as the behaviour of the models remains consistent with the previous description: the models that produced validation loss above the 2.0 mark showed less stable predictions over time when compared to other models.

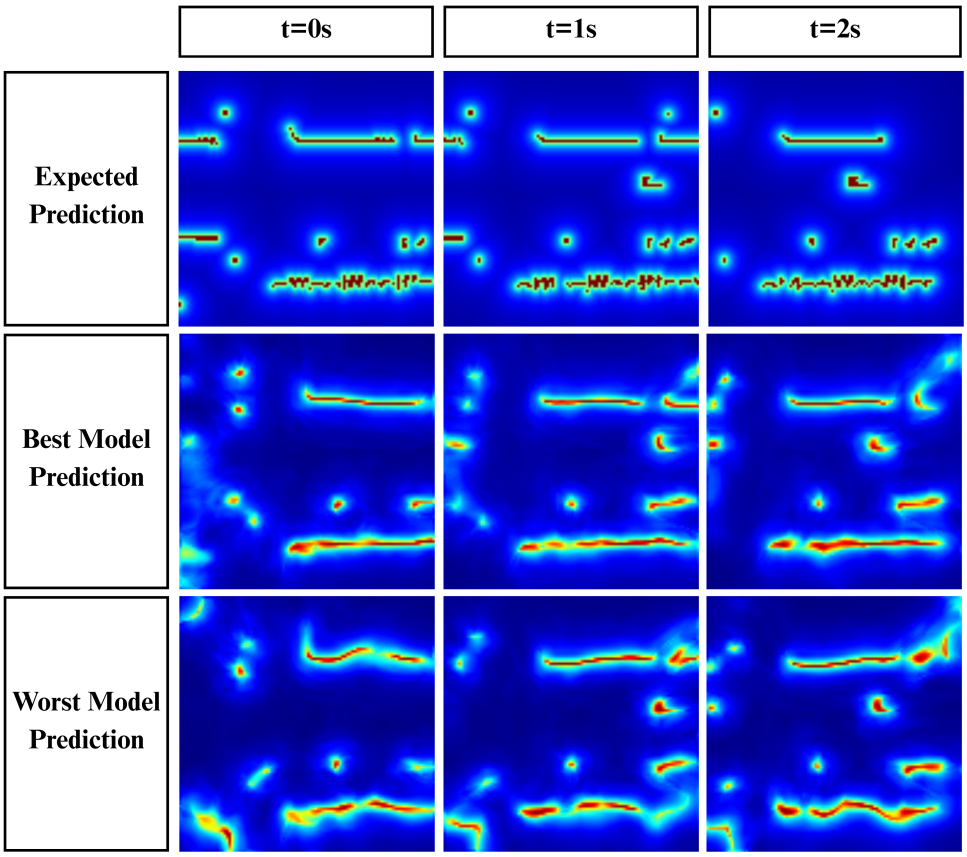


Figure 4.5: Occupancy map predictions for 3-second sequence. The best model predictions are compared against the worst model predictions. The Best model was trained with MJPEG q=16 yuvj420p compressed images, and the worst model was trained with H.265 crf=51 yuvj420p compressed images.

5 Discussion

The dataset generation process using the CARLA simulator was successful, producing synthetic sensor data and map representations required for training and evaluation. As expected, the simulated sensor readings provided precise and comprehensive information about the surrounding scene. Each pixel in the color image could be mapped onto a depth image to determine the exact distance at that specific point. Similarly, the semantic image enabled the identification of objects at any specific location within the camera frame. This level of detail allowed for the creation of stable and noiseless mapping of obstacles and occlusions in the data processing phase.

The compression experiments reveal that video compression, while introducing visible artifacts, has a limited impact on the overall performance of occupancy predictions. Even under aggressive compression settings, where effects such as blocking and blurring are visually apparent, the model managed to predict many of the structures apparent in the expected Soft Occupancy maps. Interestingly, the use of 4:2:0 color subsampling did not degrade prediction quality and, in several cases, even led to better validation loss than 4:4:4 full color resolution images. This indicates that models trained on compressed image data are robust to compression-induced quality loss and capable of extracting meaningful semantic and spatial features even when camera frames are visually degraded.

However, one limitation emerged in the form of temporal stability across frame sequences. While spatial accuracy remained satisfactory across all compression levels, models trained on less compressed inputs produced more stable and consistent results across multiple predictions. Static structures, such as walls, showed more coherent geometry that resembled the expected output in these cases. High-compression variants, however, introduced subtle inconsistencies between frames, especially in static elements, leading to less temporally consistent Soft Occupancy maps. This was reflected in slightly higher validation losses and visualized prediction outputs. This could pose challenges for other tasks, such as navigation, that depend on persistent spatial understanding. It is also worth noting that frames under higher compression could be more sensitive to motion-induced artifacts, particularly in scenes involving rapid camera rotation or vibration. However, this effect was not examined in the current experiments, as the simulation exhibited smooth driving and turning patterns, with minimal abrupt movement. The camera setup remained highly stable between frames, and vibration or aggressive motion effects were not represented in the dataset.

It is important to acknowledge that these experiments were conducted entirely within the CARLA simulation environment. CARLA provides a highly controllable and repeatable testing ground, with consistent lighting, sensor calibration, and environmental conditions. While this offers significant advantages in terms of experimental rigor and reproducibility, it also introduces limitations in terms of generalizability. Real-world datasets often involve a wider variety of camera

configurations, lighting conditions, weather variability, and sensor noise, which may interact differently with compression artefacts. Furthermore, real-world vehicle speeds and motion patterns could increase temporal instability, especially when using lossy compression schemes.

Therefore, while the results in CARLA suggest a resilience of neural reconstruction systems to compression, further validation on real-world datasets is essential. Future work could investigate the impact of compression across different physical camera setups and vehicle configurations to better understand how compression interacts with the complexities of real-world data capture. Such investigations would help narrow down the bounds of compression that can safely be applied without degrading scene understanding performance. In addition, future work could utilize the developed CARLA simulation pipeline to analyse how camera positioning affects the performance of occupancy predictions.

6 Conclusion and Future Work

This thesis explored the impact of video compression on a SimpleBEV-based occupancy prediction machine learning model. Using the CARLA simulator, a synthetic dataset was generated, and various levels of video compression were applied to RGB inputs before feeding them into the occupancy prediction model. The findings indicate that compression does not have a significantly adverse effect on the occupancy predictions. With low to moderate compression, validation performance remained around the validation loss value achieved with uncompressed images. Even with highly compressed images, which introduced clear visual artefacts and resulted in higher validation loss, the reconstructed predictions remained structurally similar to the expected output.

Importantly, the compression did influence the temporal stability of static scene elements. Lower compression rates produced smoother and more consistent reconstructions over time, whereas higher compression levels introduced jitter and instability in otherwise immobile structures. This trade-off highlights the need to consider both compression efficiency and downstream requirements when designing perception systems for real-time applications.

Given that all experimentation was conducted in a simulated environment, the generalization of these results to real-world scenarios requires further investigation. Variability in camera hardware, vehicle dynamics, and environmental conditions could yield different outcomes. Future work should explore the effects of compression on real-world datasets to better assess practical limitations and to develop adaptive approaches that maintain performance while minimizing bandwidth or storage usage.

Bibliography

[1] Encyclopædia Britannica. - Autonomous vehicle.
https://www.britannica.com/technology/autonomous-vehicle 12.05.2025.

[2] Huaiyuan Xu, Junliang Chen, Shiyu Meng, Yi Wang, Lap-Pui Chau. - A Survey on Occupancy Perception for Autonomous Driving: The Information Fusion Perspective.
https://arxiv.org/abs/2405.05173 08.05.2024.

[3] S. Akramullah, *Digital Video Compression Techniques*, in *Digital Video Concepts, Methods, and Metrics*, Apress, Berkeley, CA, 2014, pp. 3.

[4] P. Pawłowski and K. Piniarski, "Efficient Lossy Compression of Video Sequences of Automotive High-Dynamic Range Image Sensors for Advanced Driver-Assistance Systems and Autonomous Vehicles," *Electronics*, 2024, **13**(18), 3651. DOI:10.3390/electronics13183651.

[5] I. E. Richardson, *The H.264 Advanced Video Compression Standard*, 2nd ed., Wiley, 2010, pp. 25-27. DOI:10.1002/9780470989418.

[6] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An Open Urban Driving Simulator," *arXiv preprint arXiv:1711.03938*, 2017. Available: https://arxiv.org/abs/1711.03938

[7] Caesar, H., Bankiti, V., Lang, A. H., Vora, S., Liong, V. E., Xu, Q., ... Urtasun, R. - nuScenes: A multimodal dataset for autonomous driving.
https://arxiv.org/pdf/1903.11027 12.05.2025.

[8] Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., ... Sapp, B. - Scalability in Perception for Autonomous Driving: Waymo Open Dataset.
https://arxiv.org/abs/1912.04838 12.05.2025.

[9] Behley, J., Garbade, M., Milioto, A., Quenzel, J., Behnke, S., Stachniss, C., Gall, J. - SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences.
https://www.cvlibs.net/publications/Geiger2013IJRR.pdf 12.05.2025.

[10] G. Roelofs, "An Introduction to PNG," in *PNG: The Definitive Guide*, O'Reilly Media, 1999. Available: http://www.libpng.org/pub/png/book/chapter01.htmlpng.ch01.div.1

[11] A. Elfes, "Using Occupancy Grids for Mobile Robot Perception and Navigation," *Computer*, vol. 22, no. 6, pp. 46–57, June 1989. DOI: 10.1109/2.30720. Available at: ResearchGate

[12] P. Gomez-Zamora, S. Bafna, C. Zimring, E. Do, and M. R. Vega, "Spatiotemporal Occupancy for Building Analytics," in *Proceedings of the 37th Education and Research in Computer Aided Architectural Design in Europe (eCAADe) and 23rd Iberoamerican Society of Digital Graphics (SIGraDi) Joint Conference*, vol. 7, no. 1, pp. 111–120, Porto, Portugal, 2019. Available: https://www.proceedings.blucher.com.br/article-details/spatiotemporal-occupancy-for-building-analytics-34253

[13] E. M. Boczko and T. R. Young, "The Signed Distance Function: A New Tool for Binary Classification," *arXiv preprint arXiv:cs/0511105*, 2022. Available: https://arxiv.org/abs/cs/0511105

[14] J. Ortiz, A. Clegg, J. Dong, E. Sucar, D. Novotny, M. Zollhoefer, and M. Mukadam, "iSDF: Real-Time Neural Signed Distance Fields for Robot Perception," *arXiv preprint arXiv:2204.02296*, 2022. DOI:10.48550/arXiv.2204.02296.

[15] P. F. Felzenszwalb and D. P. Huttenlocher, "Distance Transforms of Sampled Functions," *Cornell University Technical Report*, 2004. Available at: https://www.cs.cornell.edu/ dph/-papers/dt.pdf

[16] Simon Green. Signed Distance Fields Using Single-Pass GPU. In Wolfgang Engel, editor, *GPU Gems 3*, chapter 34. Addison-Wesley Professional, 2007. Available from: `https://developer.nvidia.com/gpugems/gpugems3/part-v-physics-simulation/chapter-34-signed-distance-fields-using-single-pass-gpu`

[17] J. P. Richa, J.-E. Deschaud, F. Goulette, and N. Dalmasso, "UWED: Unsigned Distance Field for Accurate 3D Scene Representation and Completion," *arXiv preprint arXiv:2203.09167*, 2022. DOI:10.48550/arXiv.2203.09167.

[18] Y. Zhang, J. Zhang, Z. Wang, J. Xu, and D. Huang, "Vision-based 3D occupancy prediction in autonomous driving: a review and outlook," *arXiv preprint arXiv:2405.02595*, 2024. DOI:10.48550/arXiv.2405.02595.

[19] S. Akramullah, *Digital Video Compression Techniques*, in *Digital Video Concepts, Methods, and Metrics*, Apress, Berkeley, CA, 2014, pp. 29-34.

[20] S. Akramullah, *Digital Video Compression Techniques*, in *Digital Video Concepts, Methods, and Metrics*, Apress, Berkeley, CA, 2014, pp. 29-34.

[21] S. Akramullah, *Digital Video Compression Techniques*, in *Digital Video Concepts, Methods, and Metrics*, Apress, Berkeley, CA, 2014, pp. 39-43.

[22] FFmpeg Developers. FFmpeg [Internet]. FFmpeg; [cited 2025 May 21]. Available from: https://ffmpeg.org/

[23] FFmpeg Developers. FFmpeg Codecs Documentation [Internet]. FFmpeg; [cited 2025 May 21]. Available from: https://ffmpeg.org/ffmpeg-codecs.html

[24] FFmpeg Developers. FFmpeg MPEG-4 Encoding Guide [Internet]. FFmpeg; [cited 2025 May 21]. Available from: https://trac.ffmpeg.org/wiki/Encode/MPEG-4

[25] CARLA Developers. Traffic Manager Tutorial [Internet]. CARLA Simulator; [cited 2025 May 23]. Available from: https://carla.readthedocs.io/en/0.9.15/tuto$_{G}traffic_{m}anager/$

[26] Merriam Webster. - JPEG.
https://www.merriam-webster.com/dictionary/JPEG 19.05.2025.

[27] CARLA Simulator. - An open-source simulator for autonomous driving research.
https://carla.readthedocs.io/en/latest/start_introduction/ 12.05.2025.

[28] H. P. Moravec and A. Elfes, "High Resolution Maps from Wide Angle Sonar," in *Proceedings of the 1985 IEEE International Conference on Robotics and Automation*, vol. 2, 1985, pp. 116–121, DOI:10.1109/ROBOT.1985.1087316.

[29] J. D. Hunter, "Matplotlib: A 2D graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007. DOI:10.1109/MCSE.2007.55.

[30] Gandor, T., & Nalepa, J. - First Gradually, Then Suddenly: Understanding the Impact of Image Compression on Object Detection Using Deep Learning. Sensors, 22(3), 1104.
https://doi.org/10.3390/s22031104 12.05.2025.

[31] FFmpeg. - H.264 Video Encoding Guide.
https://trac.ffmpeg.org/wiki/Encode/H.264 12.05.2025.

[32] O'Byrne, M., Vibhoothi, M., Sugrue, M., & Kokaram, A. - Impact of Video Compression on the Performance of Object Detection Systems for Surveillance Applications. arXiv preprint arXiv:2211.05805.
https://arxiv.org/abs/2211.05805 12.05.2025.

[33] Y.-Q. Shi and H. Sun, *Image and Video Compression for Multimedia Engineering: Fundamentals, Algorithms, and Standards*, 3rd ed., CRC Press, Boca Raton, FL, 2019, pp. 16-20.

[34] Bing, B. - Next-Generation Video Coding and Streaming. Wiley, 2015. pp 51-51, 66-67, 83-84, 115-116

[35] Yueyuan Li, Wei Yuan, Songan Zhang, Weihao Yan, Qiyuan Shen, Chunxiang Wang, and Ming Yang, "Choose Your Simulator Wisely: A Review on Open-source Simulators for Autonomous Driving," *arXiv preprint arXiv:2311.11056*
https://arxiv.org/abs/2311.11056 12.05.2025.

[36] Adam W. Harley, Zhaoyuan Fang, Jie Li, Rares Ambrus, Katerina Fragkiadaki "Simple-BEV: What Really Matters for Multi-Sensor BEV Perception?" *arXiv preprint arXiv:2206.07959*, 2022.
https://arxiv.org/abs/2206.07959

[37] CARLA Agents
https://carla.readthedocs.io/en/0.9.15/adv_agents/ 12.05.2025.

[38] CARLA Town03
https://carla.readthedocs.io/en/latest/map$_{t}own03/$ 12.05.2025.

Appendix A

CARLA Simulation Client and Image Processing Code Repository

The full source code used for setting up the CARLA simulation, collecting sensor data, and generating ground truth maps—including BEV occupancy, Soft Occupancy, and visibility map—is available in the following repository:

`https://github.com/TO-autonomy/CARLA-vehicle-simulation`

The repository contains:

- Documentation for the repository, including instructions for running the simulation and data processing scripts.
- Shell script for installing the CARLA simulator and necessary dependencies.
- Shell script for running the simulation client, which includes spawning the ego vehicle, configuring sensors, and controlling the simulation flow.
- Shell script for running the data post-processing sequence for generating ground truth grids used for occupancy model training.

Researchers or practitioners seeking to reproduce the dataset generation or extend the simulation scenarios may use this codebase as a reference or foundation.

Appendix B

CARLA Driving Scenario Path

Recording synthetic sensor readings in the CARLA simulator required creating a custom driving scenario. A graphical user interface (GUI) was developed to visualize drivable roads and design a path for the ego vehicle equipped with sensors. The GUI displays all drivable areas of the selected CARLA map, allowing the user to define a vehicle path by clicking on waypoints. Once the path is created, it is saved in a JSON file. These files could later be used in simulations without further modification. During simulation, the driving agent autonomously follows the path, choosing the shortest legal route between waypoints. Additionally, a validation vehicle runs through the path to ensure it is followed accurately.

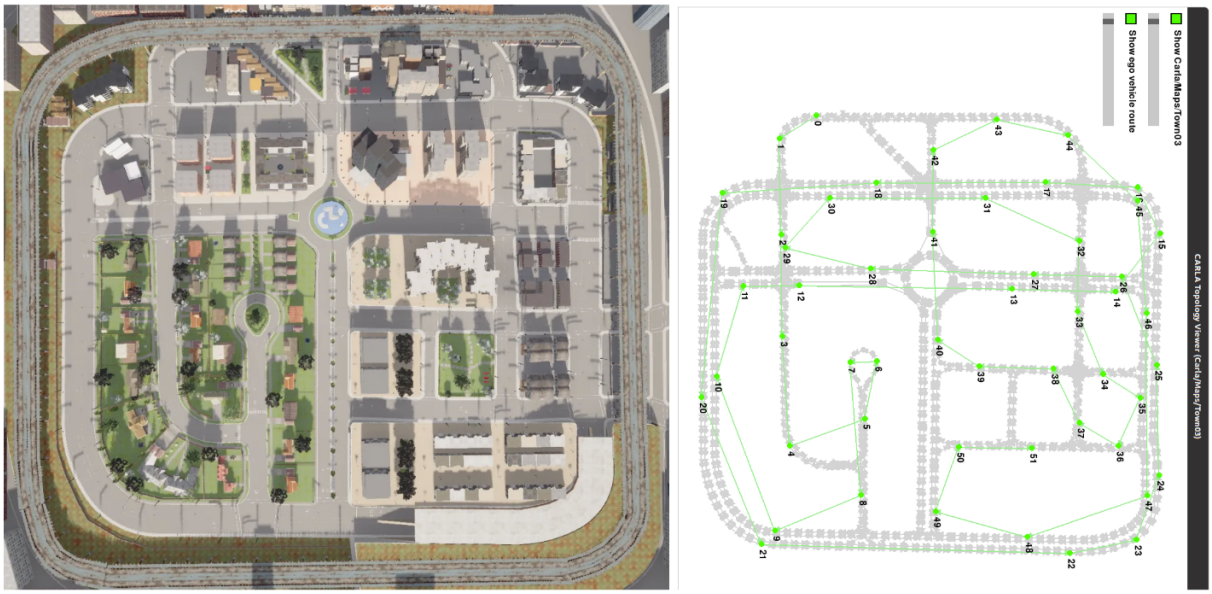


Figure B.1: A comparison between the top-down view of CARLA Town03, as shown in the official CARLA documentation [38], and the GUI developed for designing driving scenarios. The GUI visualization includes the ego vehicle's path used in the thesis experiment, beginning at node 0 and ending at node 51.

Appendix C

CARLA Synthetic Dataset

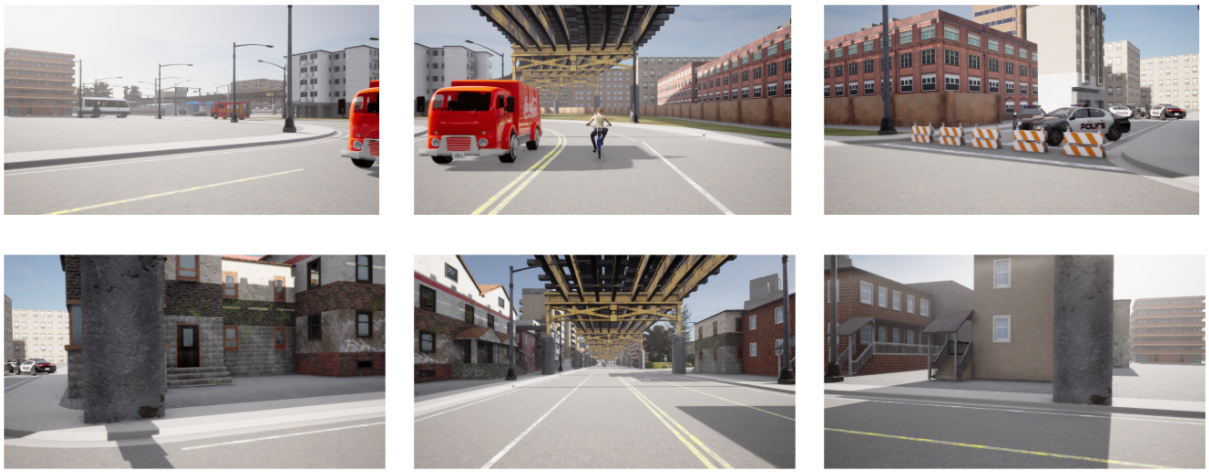


Figure C.1: Simulation vehicle view for all 6 cameras at a random timestamp.

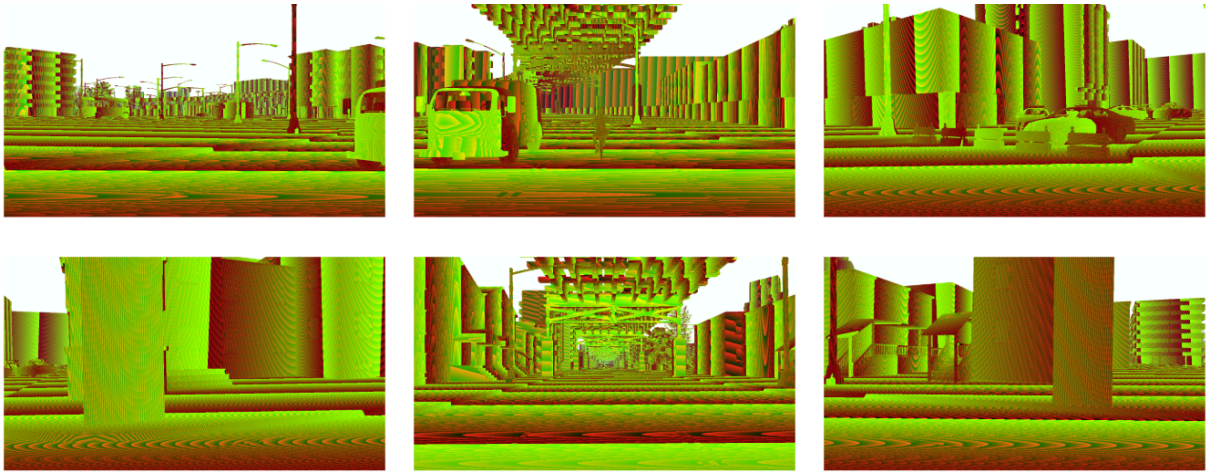


Figure C.2: Simulation vehicle view for all 6 depth cameras at a random timestamp. The depth values are encoded into each pixel's red, green, and blue channel value, which causes the striped appearance.

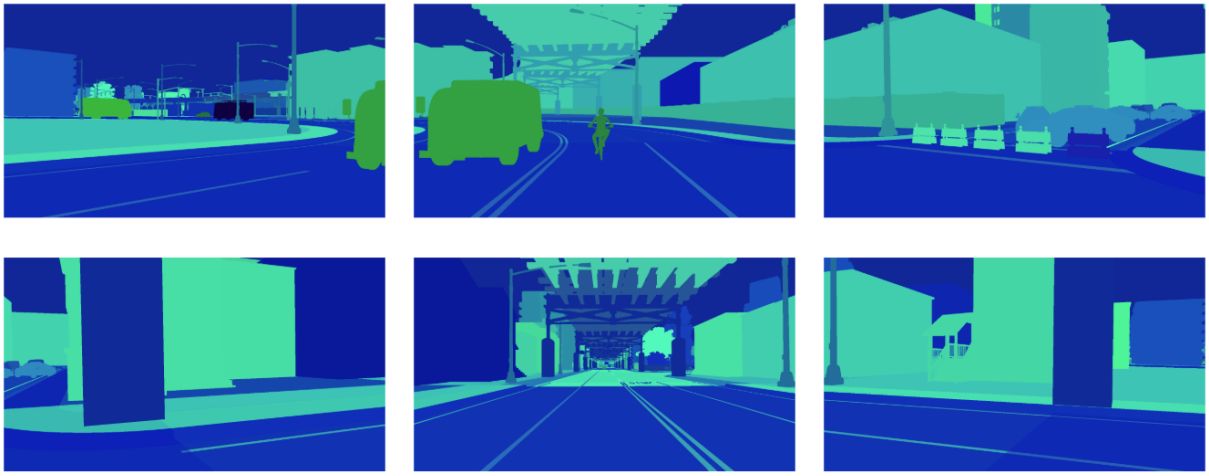


Figure C.3: Simulation vehicle view for all 6 semantic cameras at a random timestamp. The semantic and instance values for each object are encoded into every pixel's red, green, and blue channel value.

Appendix D

Video Encoding and Decoding Settings and Results

The source code for video compression pipeline that was used to apply video compression to uncompressed synthetic camera images is available in the following repository:

`https://github.com/leppsalujyrgen/video-compression-pipeline.git`

Encoding Setting	Encoded Image with color subsampling (YUV420)	Encoded Image without color subsampling (YUV444)
MJPEG Q=1		
MJPEG Q=16		
MJPEG Q=31		

Table D.1: Single frame comparison of MJPEG compression with different encoding settings

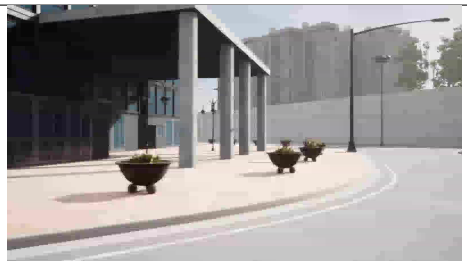
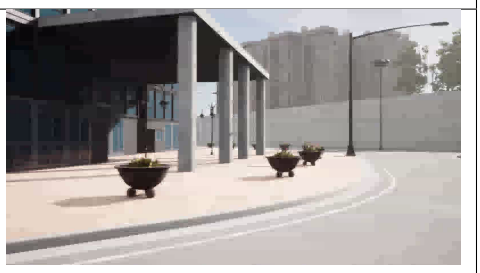
Encoding Setting	Encoded Image with color subsampling (YUV420)	Encoded Image without color subsampling (YUV444)
H.264 CRF=2		
H.264 CRF=23		
H.264 CRF=51		

Table D.2: Single frame comparison of H.264 compression with different encoding settings

Encoding Setting	Encoded Image with color subsampling (YUV420)	Encoded Image without color subsampling (YUV444)
H.265 CRF=2		N/A
H.265 CRF=23		N/A
H.265 CRF=51		N/A

Table D.3: Single frame comparison of H.265 compression with different encoding settings

Appendix E

Occupancy Network Training and Validation Results

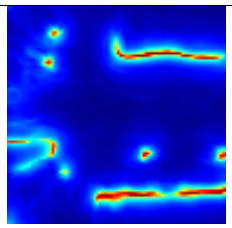
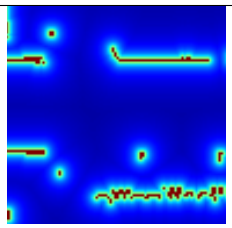
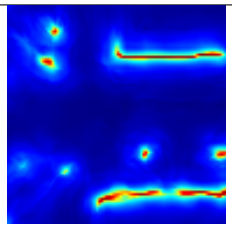
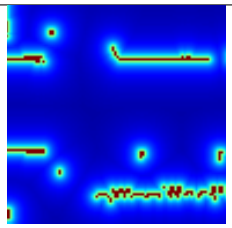
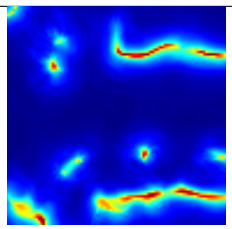
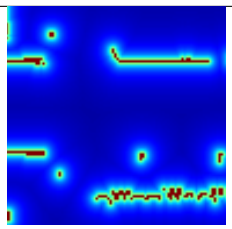
Encoding Setting	Predicted Soft Occupancy	Expected Soft Occupancy	Predicted Visibility	Expected Visibility
H.265 CRF=2 YUV420				
H.265 CRF=23 YUV420				
H.265 CRF=51 YUV420				

Table E.1: Model training results on H.265 compressed video images. The table exhibits the resulting model predictions as "Predicted Soft Occupancy" and "Predicted Visibility" columns for a selected validation timestamp. The columns "Expected Soft Occupancy" and "Expected Visibility" show the ground-truth maps generated in the data processing pipeline for comparison.

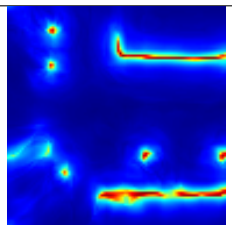
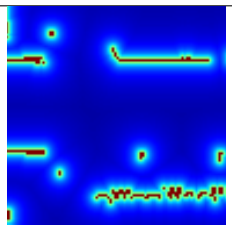
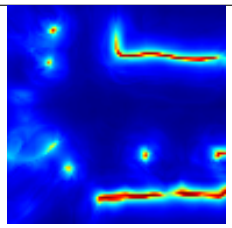
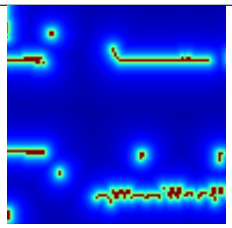
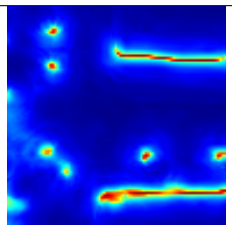
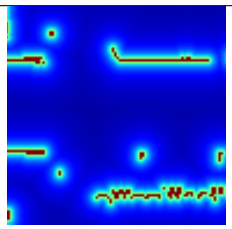
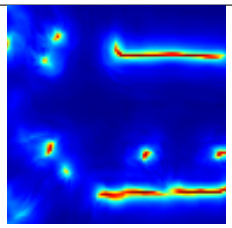
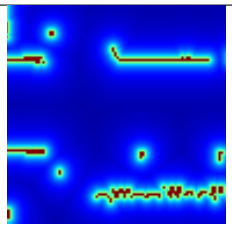
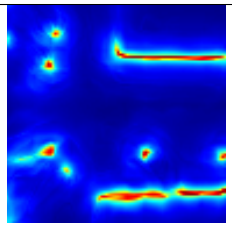
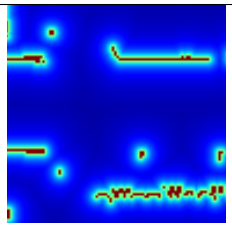
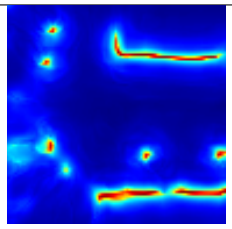
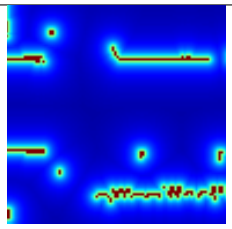
Encoding Setting	Predicted Soft Occupancy	Expected Soft Occupancy	Predicted Visibility	Expected Visibility
MJPEG Q=2 YUV420				
MJPEG Q=2 YUV444				
MJPEG Q=16 YUV420				
MJPEG Q=16 YUV444				
MJPEG Q=31 YUV420				
MJPEG Q=31 YUV444				

Table E.2: Model training results on MJPEG compressed video images. The table exhibits the resulting model predictions as "Predicted Soft Occupancy" and "Predicted Visibility" columns for a selected validation timestamp. The columns "Expected Soft Occupancy" and "Expected Visibility" show the ground-truth maps generated in the data processing pipeline for comparison purposes.

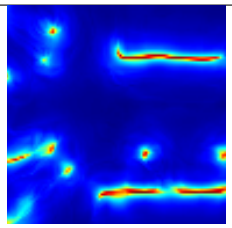
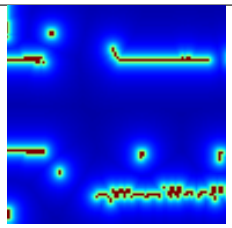
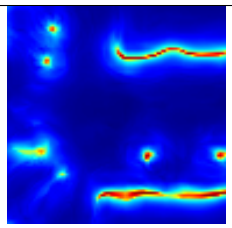
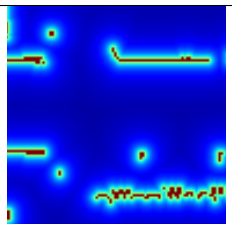
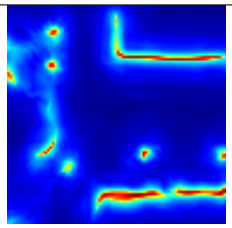
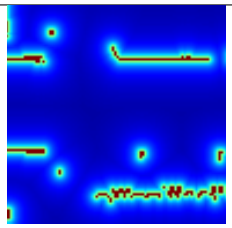
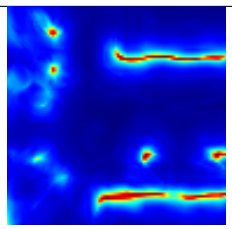
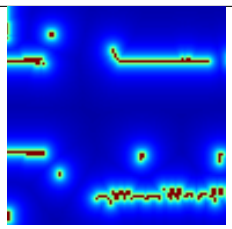
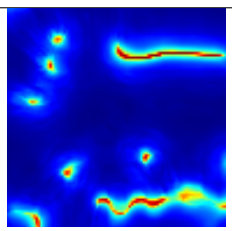
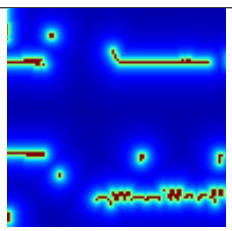
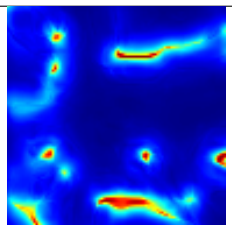
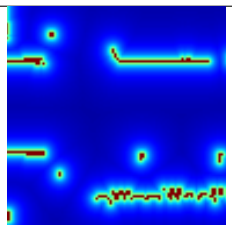
Encoding Setting	Predicted Soft Occupancy	Expected Soft Occupancy	Predicted Visibility	Expected Visibility
H.264 CRF=2 YUV420				
H.264 CRF=2 YUV444				
H.264 CRF=23 YUV420				
H.264 CRF=23 YUV444				
H.264 CRF=51 YUV420				
H.264 CRF=51 YUV444				

Table E.3: Model training results on H.264 compressed video images. The table exhibits the resulting model predictions as "Predicted Soft Occupancy" and "Predicted Visibility" columns for a selected validation timestamp. The columns "Expected Soft Occupancy" and "Expected Visibility" show the ground-truth maps generated in the data processing pipeline for comparison.

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, Jürgen Leppsalu

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose

   “Video compression effect on a camera-based occupancy prediction model”

   mille juhendaja on Rando Avarmaa

   reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace’i kaudu Creative Commonsi litsentsiga CC BY NC ND 3.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.

3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.

4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Jürgen Leppsalu
23.05.2025