Road Detection with Semantic Segmentation

Ian Quan
quanian031229@gmail.com

Abstract

This report details the development and evaluation of a road segmentation model using the U-Net architecture, tailored for autonomous driving applications. Leveraging the KITTI Vision Benchmark Suite, the model is trained from scratch on a dataset comprising 135 stereo image pairs. This study introduces an innovative approach by fusing depth maps with road ground truth to enhance segmentation accuracy. The model's performance is assessed through key metrics such as Pixel-Accuracy, Intersection over Union (IoU), and Dice Coefficient, showing significant improvement when incorporating fused depth information. The report highlights the challenges of segmenting roads under varying conditions and proposes solutions to enhance model robustness and reliability in real-world scenarios.

1 Introduction

In the rapidly evolving field of autonomous driving, precise road segmentation is crucial for the safe navigation of vehicles. This project aims to develop a road segmentation model that can accurately classify road boundaries under diverse environmental conditions. The model is built using the U-Net architecture, renowned for its efficiency in handling segmentation tasks with high precision. This architecture is particularly suited for the limited dataset size typically available in autonomous driving research, such as the KITTI Vision Benchmark Suite used in this study.

The dataset consists of high-resolution stereo image pairs, each accompanied by detailed calibration and ground truth data, facilitating the extraction of spatial depth information crucial for this task. The model training is performed from scratch, emphasizing the adaptation of the model to the specific characteristics of road surfaces captured in the stereo imagery. By integrating depth data through a fusion with the road ground truth, the model gains enhanced capabilities to discern road from non-road areas, aiming to overcome common segmentation challenges such as varying lane markings and complex traffic scenarios.

2 Stereo Image Disparity and Depth

2.1 Data

The dataset employed for this project is sourced from the KITTI Vision Benchmark Suite, specifically designed for autonomous driving research. The KITTI dataset is widely recognized for its rich set of high-resolution stereo image pairs captured in urban environments. The dataset includes several categories of data, but for the purpose of this project, we focus on the stereo and calibration data, which include:

1. **Stereo Images:** Pairs of synchronized left and right images taken from a stereo camera mounted on a moving vehicle. The high resolution provides detailed pixel-level data necessary for accurate depth estimation. The stereo camera system used in KITTI consists of two camera. The cameras are mounted on the vehicle's roof with a baseline of approximately 0.54 meters between them. These stereo images are primarily used for extracting disparity and computing depth.

2. **Calibration Files:** Each stereo pair in the dataset comes with associated calibration files. These files contain intrinsic parameters of the cameras (focal length and optical centers) and extrinsic parameters detailing the relative positions and orientations of the stereo cameras. These parameters are crucial for transforming disparity values into depth measurements. The calibration file also contains information of the baseline distance, which is the distance between the centers of the two camera lenses, is used to calculate the depth from the disparity. The focal length and baseline are extracted using the provided calibration files for each image pair.

2.2 Compute Disparity Between Two Stereo Images

2.2.1 Algorithm

The algorithm employed for disparity computation between stereo images is the Stereo Block Matching (SBM) algorithm, facilitated by OpenCV's `cv2.StereoBM_create()` function. This method is a widely-used technique in stereo vision to estimate disparity maps, which represent the pixel distance between corresponding points in left and right stereo images.

Block Matching Process: The fundamental concept of SBM involves sliding a block (a small window of pixels) from the left image across a corresponding search range in the right image and finding the best match using a cost function. The disparity for each pixel is calculated based on the horizontal shift between the matching blocks that results in the minimum cost, typically measured in terms of pixel intensity differences.

2.2.2 Parameters

1. **numDisparities:** This parameter defines the maximum disparity difference that the algorithm will search. It essentially sets the range of horizontal shifts and must be divisible by 16. A higher value increases the search range but also the computational complexity.
2. **blockSize:** Represents the size of the square block (window) that the algorithm uses to compare segments of the two images. A larger block size might improve the robustness of the disparity estimation, especially in texture-less regions, but can also smooth out fine details.

To find the best combination of numDisparities and blockSize, disparity maps were computed and visualized. These experiments helped in understanding the trade-offs between computational efficiency and disparity map quality, allowing for an informed selection of parameters best suited for the given dataset. In this project numDisparites=96 and blockSize=21 were selected.

For objects closer to the camera, the distance between their projections on the left and right images increases, resulting in a larger disparity. These objects appear with higher intensity on the disparity map. Conversely, objects that are further from the camera project are closer together on the left and right images, yielding smaller disparity values with darker areas.

2.3 Depth of each pixel

Extracting depth can be done in varying forms depending on the given sensor and data. For monocular camera, directly estimating the depth of a still image is not straightforward without external aids, such as employing supervised deep learning models or utilizing parallax techniques. In contrast, stereo vision cameras provide another method for depth extraction.

2.3.1 Calculation of depth

As depicted in Figure 2, stereo vision involves projecting a point in 3D space onto two different frames, which appear differently due to the cameras' disparate positions. Knowing the relative positions of these points allows for the calculation of the depth to the real-world point. The depth is calculated using the geometry of stereo vision. The formula to compute the depth Z of each pixel in a stereo image pair is given by:

$$Z = \frac{f \times b}{d} \tag{1}$$

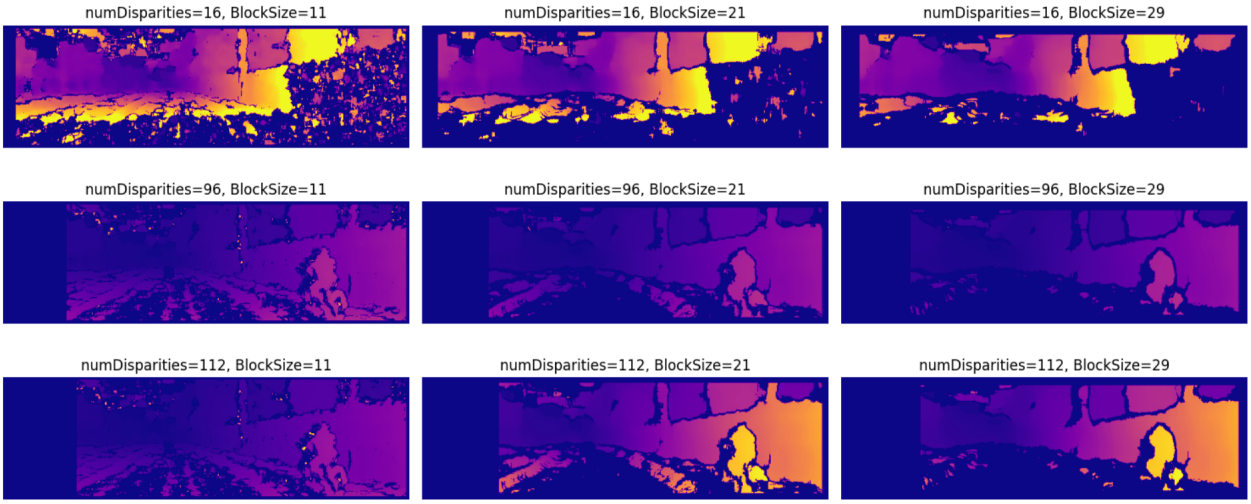

Figure 1: Disparity Map with different block size and maximum disparity difference

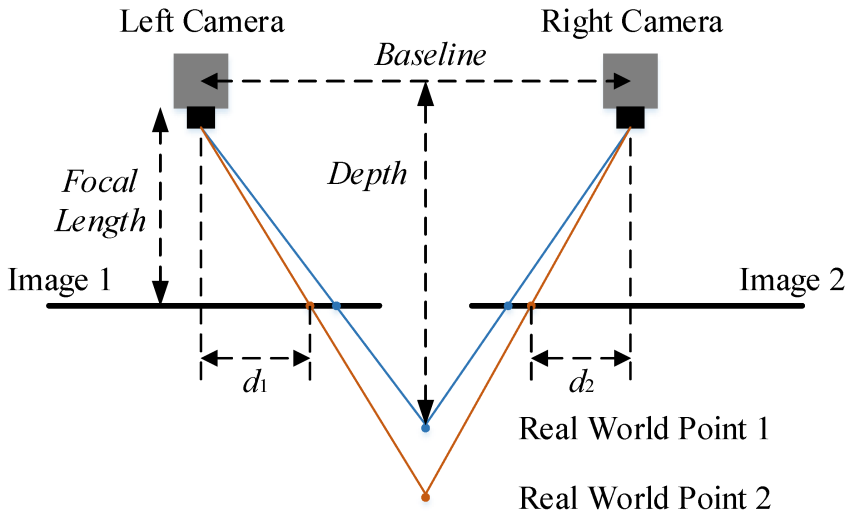

Figure 2: A basic model of stereo cameras with intrinsic camera parameters, from Liu et al. [2016]

where f is the focal length of the camera (in pixels), b is the baseline distance between the two cameras (in meters), d is the disparity (in pixels) at the pixel of interest.

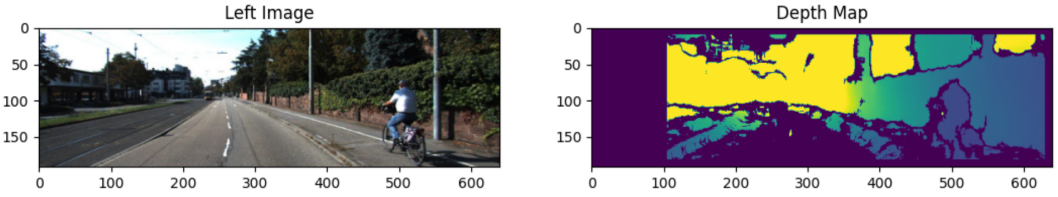

Figure 3: Original image to the left and depth map to the right. Pixels with higher intensity indicates a larger depth and vice versa

3 Road Classification

3.1 Data Processing

For this project, the KITTI Vision Benchmark Suite is employed, specifically utilizing a subset of its comprehensive data collections tailored for autonomous driving applications. The dataset consists of 135 stereo image pairs, each accompanied by ground truth annotations. High-resolution stereo images are essential for extracting depth information and understanding the spatial layout of the surroundings.

3.1.1 Pre-processing the Data and Data Splitting

The initial step in preparing the dataset for the road classification task involves several key processes aimed at optimizing the input data for effective model training. The pre-processing routine includes:

Normalization: All image inputs and depth maps are normalized to a range between 0 and 1. This step is critical as it facilitates faster convergence during training and maintains numerical stability.

Mask Normalization: The ground truth masks, which identify road areas, are converted to a floating-point format and normalized, ensuring that mask values are either 0 or 1. This binary format directly corresponds to the presence or absence of the road.

After pre-processing, the dataset is split into training, validation, and testing sets. Using an 80-20 split, 80% of the data is reserved for training and validation, while the remaining 20% is set aside for final testing. This separation helps in evaluating the model on unseen data, providing a measure of its generalization capability. Further, the training-validation portion undergoes another split where 75% is used for training and 25% for validation purposes. This approach ensures that the model is not only learning effectively but also tuning its parameters to avoid overfitting.

3.1.2 Fusing Depth Map and Road Ground Truth

To enhance the model's ability to discern road surfaces from other objects and terrain, depth maps and road masks are fused. This fusion process involves:

Depth Inversion: The normalized depth map is inverted (1 - depth value), placing nearer objects (typically higher in depth) lower in intensity which aids in distinguishing them from the road surface.

Fuse Inverted Depth Map With Ground Truth: To create a new fused ground truth, matrix multiplication was performed between the depth map and original ground truth. As shown in Equation 2. Considering that the original ground truth only contains the values 0, or 1, i.e. background or road, the zero background values will eliminate (set each pixel to zero) the irrelevant depth. While the road mask (pixel value of 1) will yield that all the road pixels for each image retain their depth values. The fusion ensures that depth information is only retained for the road areas, significantly reducing background noise and focusing the model's learning on road features.

$$a_{ij} \cdot b_{ij} = \begin{pmatrix} a_{11} \cdot b_{11} & \cdots & a_{1n} \cdot b_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} \cdot b_{m1} & \cdots & a_{mn} \cdot b_{mn} \end{pmatrix}, \quad i = 1, 2, \ldots, n, \quad j = 1, 2, \ldots, m \tag{2}$$

a_{ij} is the ground truth with ones and zeros, and b_{ij} corresponds to the inverted depth map

This fused depth map serves as the target output (labels) for the training process. An illustration of the fused depth map from an example image can be seen in Figure 4

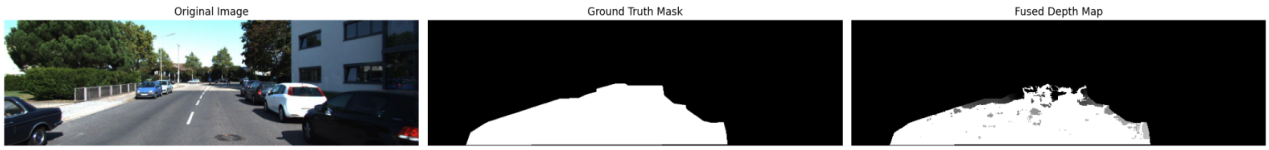

Figure 4: Fused ground truth with depth information

3.1.3 Data Augmentation

Given the limited size of the dataset, data augmentation is particularly crucial. It effectively enlarges the training dataset, providing the model with a broader range of scenarios and features to learn from.

Data augmentation artificially add variation to the dataset by introducing random, realistic transformations. For this project, horizontal flipping augmentation is applied. Images and their corresponding masks are flipped horizontally, simulating the reversal of driving lanes, which is particularly useful for ensuring the model's effectiveness across different driving contexts.

Both the image and mask data are augmented in sync using the same random seed to ensure that the transformations correspond exactly, preserving the integrity of the ground truth data. The augmented data is generated in batches, using a predefined batch size to optimize memory usage and computational efficiency during training. This yielded a less risk of overfitting and better generalization of the model while extending the data with varying training images.

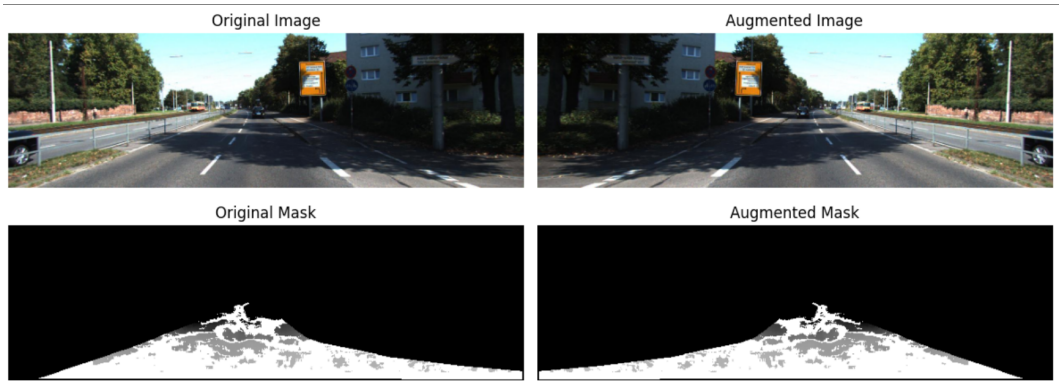

Figure 5: Example image and mask that is flipped for data augmentation

3.2 U-Net

3.2.1 Model Architecture

The U-Net architecture is a convolutional neural network originally designed for biomedical image segmentation. Its structure is particularly well-suited for tasks requiring precise localization, making it ideal for road segmentation in stereo images. The architecture can be divided into two main components: the encoder (contracting path) and the decoder (expansive path), connected by a bridge.

Encoder: The encoder consists of a series of blocks, each containing two convolutional layers followed by a batch normalization layer and a ReLU activation. These blocks are designed to capture the underlying features of the image at various resolutions and contexts. After each convolutional block, a max-pooling operation is performed, reducing the spatial dimensions by half and incrementally increasing the depth of feature maps. This process helps the network focus on the most salient features, enhancing its ability to generalize across different visual scenarios.

Decoder: The decoder mirrors the encoder's structure but inversely; it gradually upscales the feature maps to reconstruct the original image dimensions. Each step in the decoder begins with a transposed convolution, effectively doubling the size of the feature maps. These are then concatenated with the

corresponding encoded feature maps preserved from the encoder through skip connections. These connections are crucial as they reintroduce the spatial details lost during downscaling, allowing for precise localization and detailed segmentation.

Output Layer: The final layer of the network is a convolutional layer with a sigmoid activation function, which maps the final feature representation to the desired output format. In the case of road segmentation, the output is a binary mask that classifies each pixel as road or non-road.

The U-Net architecture can be visualized as a U-shaped structure, where the bottom of the "U" represents the bridge, and the two arms are the encoder and decoder. This design facilitates the flow and refinement of feature maps throughout the network, ensuring detailed and accurate segmentation outputs. With the U-Net, we can solve the two questions of segmentation: “what” and “where.”

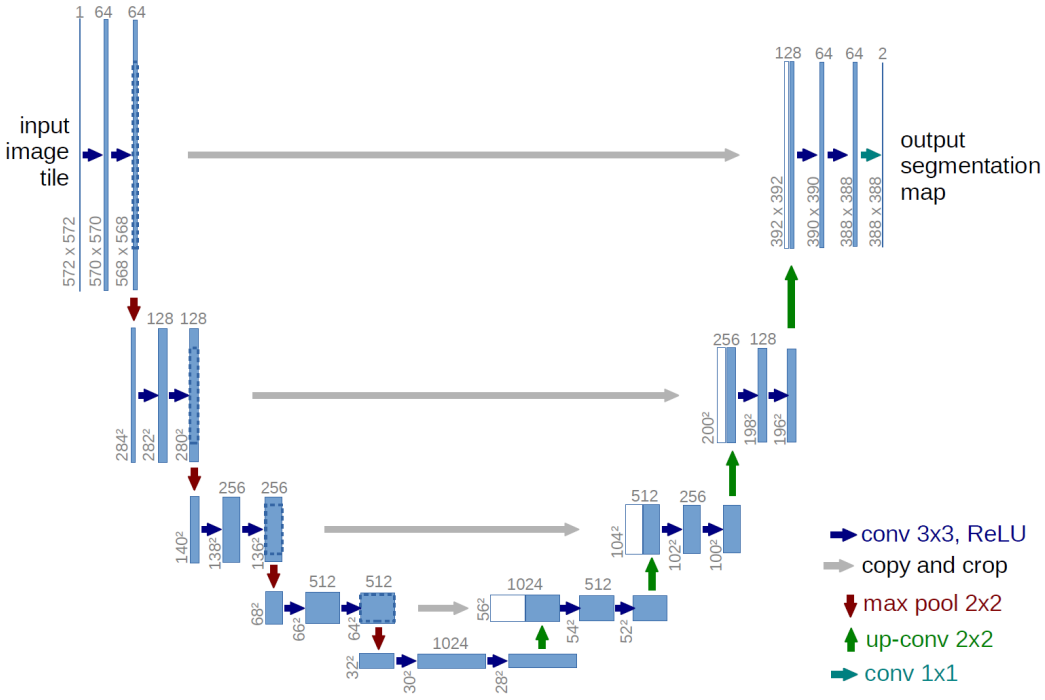

Figure 6: U-Net Architecture from Ronneberger et al. [2015]

3.2.2 Rationale Behind the Choice of using U-Net for Road Semantic Segmentation

This decision is driven by several key factors that make U-Net particularly suitable for this application:

High Precision Localization: U-Net excels in tasks where precise localization is crucial. For road segmentation, accurately delineating the boundaries of the road is important for safe navigation of autonomous vehicles. U-Net’s architecture, with its contracting and expansive paths, ensures that both high-level context and fine-grained detail are captured and utilized effectively.

Efficiency with Limited Data: Originally developed for medical image segmentation where datasets are typically small, U-Net is designed to perform well with limited data through its use of extensive data augmentation and efficient use of image data via skip connections. This capability is particularly advantageous given the limited size of the KITTI dataset used in our project.

Skip Connections: These are integral to U-Net and allow the transfer of fine-grained details from the encoder to the decoder network. In road segmentation, these connections help preserve important spatial information (like edge details of the road) that might be lost during the pooling operations in the encoder. By reintegrating this information during the decoding phase, U-Net ensures more accurate and detailed segmentation results.

End-to-End Learning: Unlike systems that rely on hand-engineered features or multiple stages of processing, U-Net provides a straight-forward, end-to-end learning solution that directly outputs the segmentation mask from the raw input images. This simplifies the modeling pipeline and potentially reduces the time to deployment.

3.2.3 Model Evaluation

In assessing the performance of the U-Net model for road segmentation, some evaluation metrics are chosen to measure the performance of the neural networks. The metrics chosen are among several that are the most suitable for this evaluation purpose, used by many state of the art research paper within the same field. The primary metrics used for evaluating our U-Net model include:

Dice Coefficient: This metric, also known as the Sørensen–Dice coefficient, is a statistical tool that measures the similarity between two samples. It is particularly well-suited for evaluating models on segmentation tasks because it compares the pixel-wise agreement between the predicted segmentation masks and the true masks. The Dice coefficient is defined as:

$$\text{Dice Coefficient}(\hat{y}, y) = \frac{2\text{TP}(\hat{y}_i, y_i)}{2\text{TP}(\hat{y}_i, y_i) + \text{FP}(\hat{y}_i, y_i) + \text{FN}(\hat{y}_i, y_i)} \tag{3}$$

where $\hat{y}_i$ is the predicted value of the pixel and y_i being the ground truth value of the labeled pixel i

Intersection over Union (IoU): Also known as the Jaccard index, this metric measures the overlap between the predicted mask and the true mask. The IoU is calculated by dividing the area of overlap between the predicted and actual masks by the area of union between them. Like the Dice coefficient, a higher IoU score indicates better segmentation accuracy. This metric is especially critical for tasks where boundaries and precise delineation are important, such as road segmentation. The Intersection over Union is defined as:

$$\text{IoU}(\hat{y}, y) = \frac{\text{TP}(\hat{y}_i, y_i)}{\text{TP}(\hat{y}_i, y_i) + \text{FP}(\hat{y}_i, y_i) + \text{FN}(\hat{y}_i, y_i)} \tag{4}$$

where $\hat{y}_i$ is the predicted value of the pixel and y_i being the ground truth value of the labeled pixel i

Accuracy: This is a straightforward metric that measures the proportion of correctly identified pixels (both road and non-road) in the segmentation mask. While accuracy is a useful metric, it can sometimes be misleading, especially in cases where there is a significant class imbalance (e.g., much more non-road area than road). That is why our model's loss function is based on the negative Dice coefficient instead of accuracy.

$$\text{Accuracy}(\hat{y}, y) = \frac{\text{Matched Predictions}}{N}, \quad \text{Matched Predictions} = \sum_{i=1}^{N} \begin{cases} 1, & \hat{y}_i = y_i \\ 0, & \hat{y}_i \neq y_i \end{cases} \tag{5}$$

Choice of Dice Coefficient for Loss Function

The decision to use the Dice coefficient for the loss function, rather than solely relying on accuracy, stems from its effectiveness in handling the class imbalance typically found in road segmentation tasks. The Dice loss (defined as 1 - Dice Coefficient) helps to directly optimize the model towards increasing the overlap between the predicted and ground truth masks, ensuring that both the presence and precise localization of roads are accurately captured. The choice of Dice coefficient for the loss function promotes better generalization across varying scenes and lighting conditions, ensuring the model remains robust and reliable under different operational environments.

3.2.4 Model Training and Model Parameters

Training Process

The training process of a U-Net model begins with initializing the U-Net architecture, compiling it with the Adam optimizer and Dice coefficient loss. Throughout the training, the model monitors several key metrics: traditional accuracy, the Dice coefficient itself, and the Intersection over Union (IoU). These metrics provide a comprehensive view of the model's performance, emphasizing both general accuracy and the specific quality of the segmentation. To prevent overfitting and enhance training efficiency, an early stopping callback is implemented, which halts training if there is no improvement in the validation Dice coefficient for five consecutive epochs.

Model Parameters

The key parameters of the U-Net model include:

1. **Input Shape:** The input resolution, as well as the output resolution for all images were 640×192. The input was however a RGB image while the output was in gray-scale as mentioned.
2. **Optimizer:** ADAM optimizer with a initial learning rate of 0.001.
3. **Loss Function:** Negative Dice coefficient, optimizing for overlap between predicted and true segmentation.
4. **Batch Size:** Set to 8 to manage computational resources effectively while allowing for sufficient gradient approximation.
5. **Epochs:** 100, with early stopping based on validation performance to prevent overfitting. The number of epochs is larger than normal due to the lack of training data.

3.3 Results of Road Classification

The performance of the U-Net model for road segmentation was evaluated using key metrics: Pixel-Accuracy, Intersection over Union (IoU), and Dice Coefficient. We compared the results of a standard U-Net model with a variant where a fused mask incorporating 3D features was used. The detail plots of the three metrics can be viewed in the appendix on table 12, table 13 and table 14.

From table 1, it clearly shows that the U-Net model with the fused mask significantly outperforms the standard U-Net model across all metrics. The improvements observed with the addition of the fused depth mask, underline the benefit of integrating additional contextual and depth-related information into the segmentation process. The use of 3D features allows the model to better understand the spatial relationships and structures within the images, leading to more accurate segmentation results. This approach is particularly effective for complex environments where depth cues play a critical role in distinguishing between different elements within the scene.

Table 1: Road Classification Results

Model	Pixel-Accuracy	IoU	Dice-Coefficient
U-Net	0.9486	0.8127	0.8965
U-Net with fused mask	0.9611	0.8366	0.9099

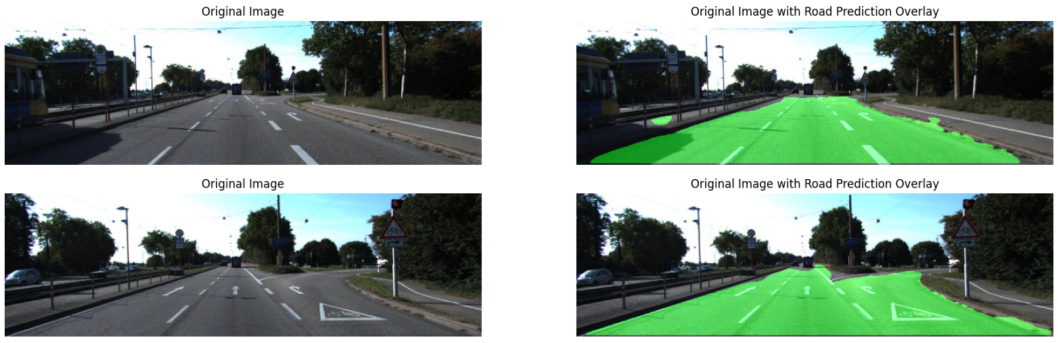

Figure 7: U-Net model result

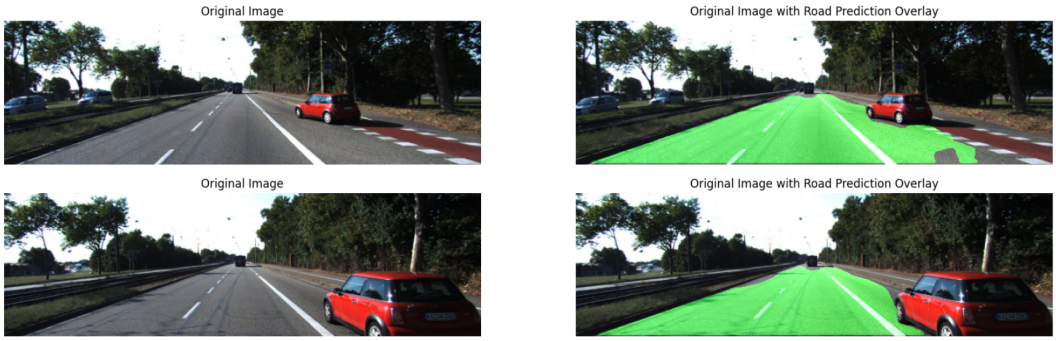

Figure 8: U-Net model result

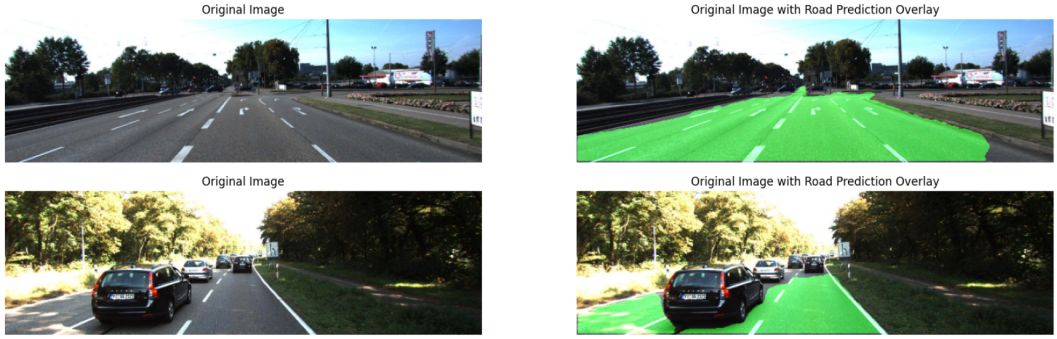

Figure 9: U-Net model result

3.4 Challenges and Limitaions

While the U-Net model has demonstrated considerable success in segmenting road areas from stereo images, there are inherent challenges and limitations that affect its performance. These challenges are particularly evident in complex traffic scenarios and under specific road conditions.

3.4.1 Classification in the Presence of Lane Markings

One of the significant challenges encountered is the model's performance around lane markings, especially when there are distinct lines separating lanes. The presence of these lines can confuse the model, leading to inaccurate classifications where parts of the road are misclassified as non-road areas. This issue stems from the model's sensitivity to high-contrast features, which can overpower the more subtle cues that define continuous road surfaces.

Solution: Expand the dataset to include a wider variety of road conditions, including various types of lane markings and more diverse traffic scenarios. More extensive data collection efforts could help in training models that are more adaptive and robust to the complexities of real-world driving.

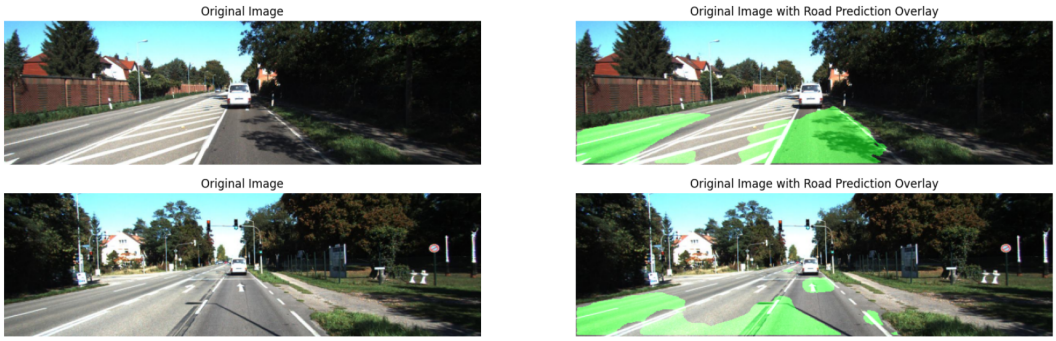

Figure 10: Example of misclassified road pixels in lane marking condition

3.4.2 Insufficient Training Data

Another critical limitation is the amount of training data available. The model's ability to generalize well across various road conditions heavily relies on the diversity and volume of data used during training. Currently, the dataset, while robust in many respects, does not comprehensively cover all possible variations in road conditions, weather scenarios, and traffic densities. Limited data particularly affects the model's performance in less common but crucial scenarios, such as roads with atypical lane markings, temporary road signs, or unusual weather conditions. This limitation can lead to a lack of robustness in the model, where it performs well under normal conditions but fails under less typical or more challenging conditions.

4 Plane Fitting for Road Surface in 3D point Cloud

4.1 Methodology

Fitting a plane to the road surface within a 3D point cloud comprised several key steps:

4.1.1 3D Coordinate Extraction

For each pixel identified as part of the road from the output of the road classification model, its 3D coordinates were calculated using the depth map generated from stereo images using equation 1. The depth map provides the distance from the camera to each pixel, while the camera's intrinsic parameters (focal length and principal points) allow for the conversion of these depth values into 3D coordinates (X, Y, Z).

Given:

- Z: Depth of the pixel from the camera, obtained from the depth map with equation 1
- p_x, p_y: Principal points of the camera, obtained from the calibration files.
- f: Focal length of the camera.
- u, v: Original pixel coordinates in the image's 2D space.

The 3D coordinates (X, Y, Z) are computed as follows:

$$X = \frac{(u - p_x) \times Z}{f} \tag{6}$$

Here, $(u - p_x)$ calculates the horizontal displacement of the pixel from the principal point along the x-axis, normalized by the focal length f. This value is then scaled by the depth Z to convert it into a real-world horizontal distance from the center of the camera.

$$Y = \frac{(v - p_y) \times Z}{f} \tag{7}$$

Similarly, this applies to the vertical plane.

With equation 6 and 7, the coordinates can be compactly expressed in vector form as:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = Z \times \begin{bmatrix} \frac{(u-p_x)}{f} \\ \frac{(v-p_y)}{f} \\ 1 \end{bmatrix} \tag{8}$$

For this project, the principal points p_x, p_y, focal length and baseline f can be extracted from the calibration data of the cameras used in the KITTI dataset. The values are as follows: Principal Point X: 609.5593, Principal Point Y: 172.8540, Focal Length: 721.5377 pixels, Baseline: 0.5372 meters.

4.1.2 Robust Plane Fitting:

To accurately model the road surface and be robust against outliers (which are common in real-world data due to various factors like shadows, occlusions, and non-uniform road textures), a RANSAC (Random Sample Consensus) algorithm was employed. This algorithm iteratively fits multiple models to subsets of the data, estimating the parameters of a plane that best fits the road points while ignoring outliers.

4.1.3 Visualization:

Point Cloud Visualization: The 3D coordinates of road pixels were plotted as a point cloud to provide a visual context of the data structure and distribution. Plane Visualization: Alongside the point cloud, the plane estimated by the RANSAC algorithm was plotted. This involved creating a grid of (X, Y) coordinates that span the extent of the observed data and using the plane model to predict the Z values, forming a visual representation of the estimated road plane.

4.2 Visualization

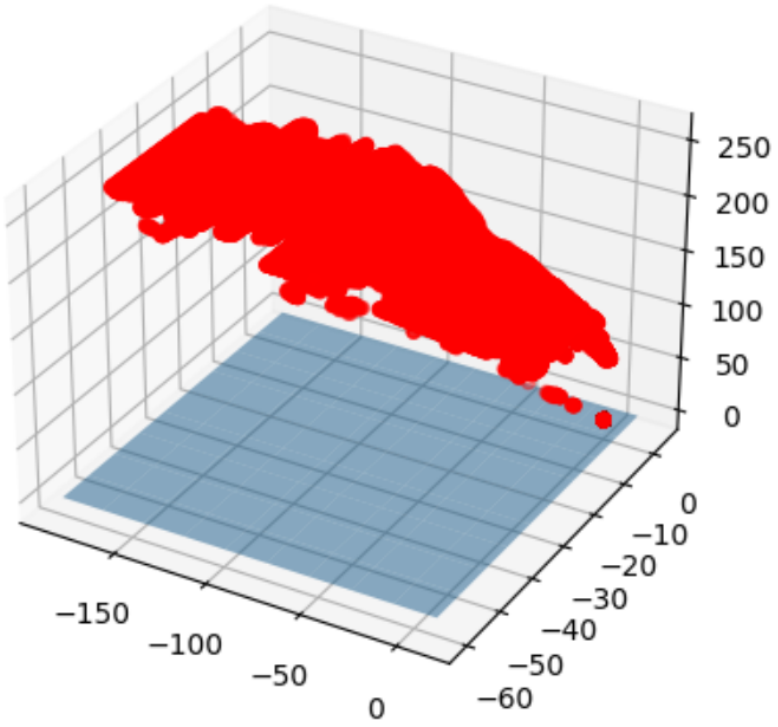

Figure 11: 3D point cloud of the road pixels with the estimated ground plane

Appendix

Below are the metrics used to evaluate the performance of the U-Net model for road classification. Blue label is the base line U-Net model, while orange label is U-Net with fused depth mask.

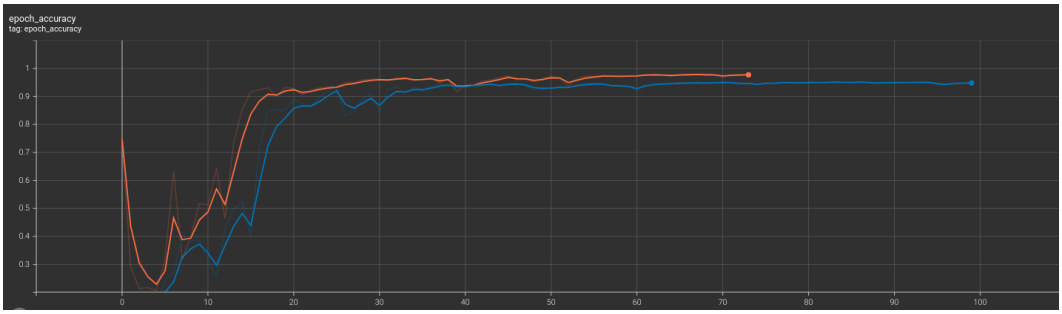

Figure 12: Evaluation accuracy of U-Net model for road classification

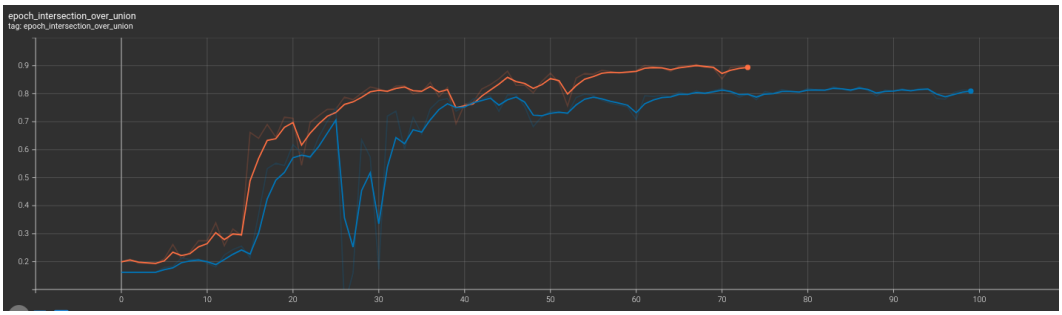

Figure 13: IoU of U-Net model for road classification

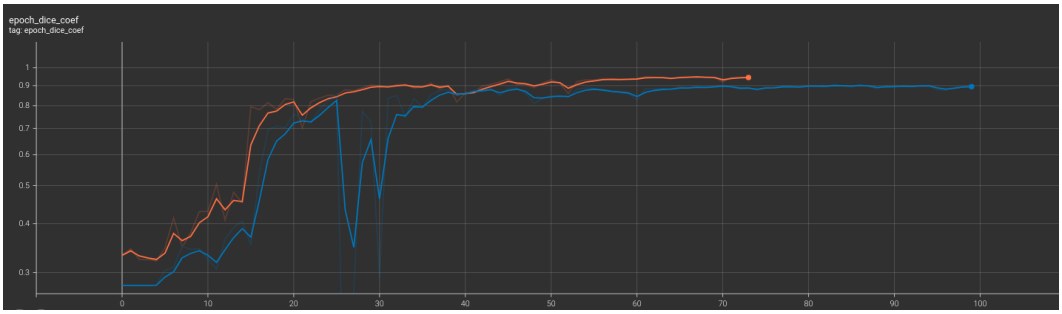

Figure 14: Dice Coefficient of U-Net model for road classification

References

Qiang Liu, Ruihao Li, Huosheng Hu, and Dongbing Gu. Extracting semantic information from visual data: A survey. *Robotics*, 5(1), 2016. ISSN 2218-6581. doi: 10.3390/robotics5010008. URL `https://www.mdpi.com/2218-6581/5/1/8`.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015. URL `http://arxiv.org/abs/1505.04597`.