Report and Result - Road Semantic Segmentation

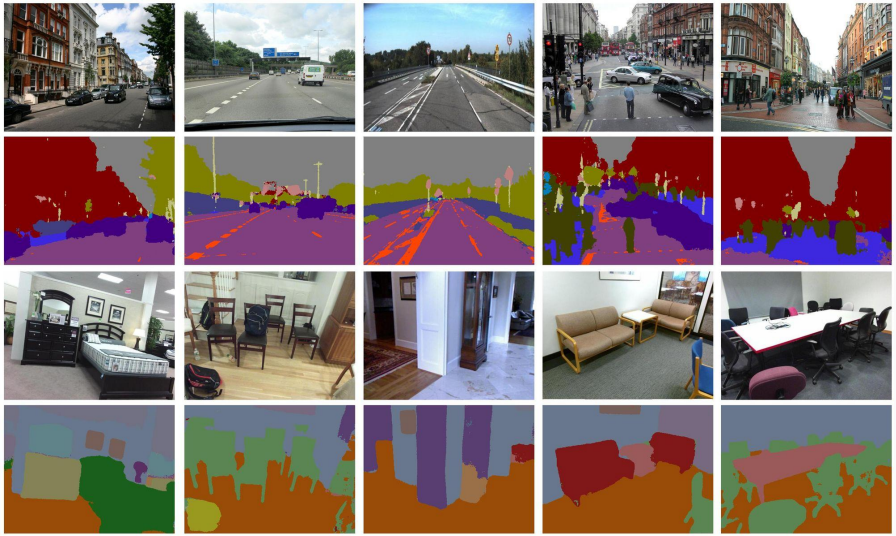
BY : Eshaan Agarwal

Content :

- Introduction
- My approach
- Model Architecture
- Why This Model ?
- Training Parameters
- Model Description
- Results

Introduction : Semantic Segmentation

Semantic segmentation :- Semantic segmentation is the process of classifying each pixel belonging to a particular label. It doesn't different across different instances of the same object. For example if there are 2 cats in an image, semantic segmentation gives same label to all the pixels of both cats.



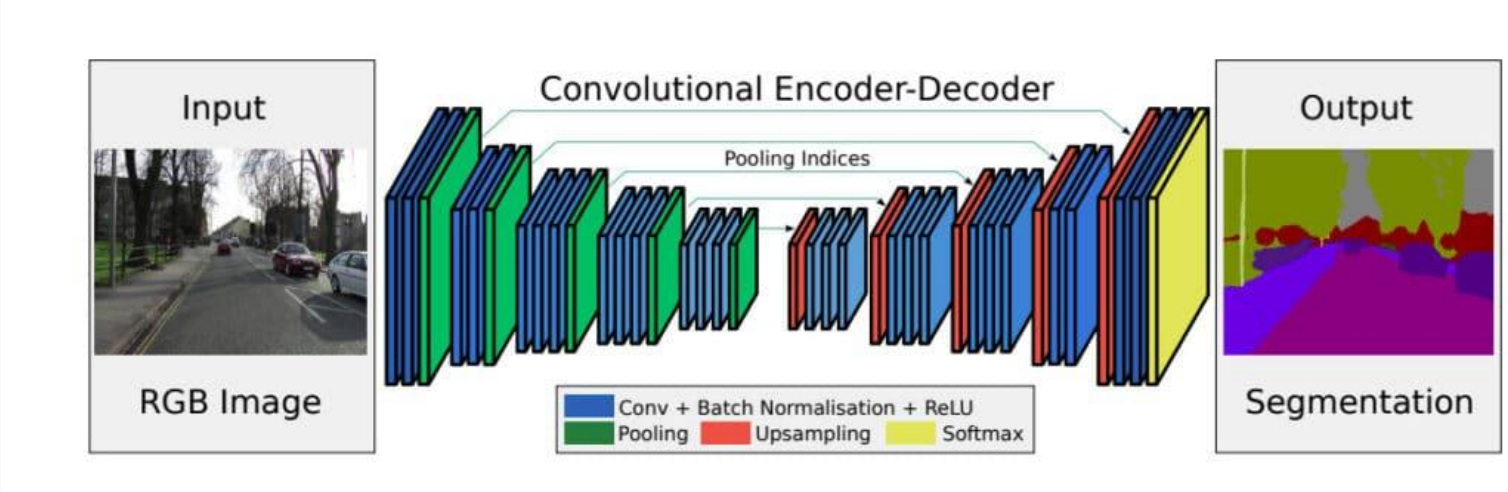
Different Usecases : Autonomous vehicles and Medical Diagnosis

My Approach : U-net and MobileVnetU2

In my approach i applied the following - U-net. They were first implemented from scratch and Since i was ill due to covid I could only train U-net.

Augmentation Techniques used to make a reliable dataset for training and efficient performance.

Training Done :
The KITTI Road dataset is utilised for training and testing the algorithms on road segmentation.
Garden and Agro Dataset



U-Net and MobileNetV2

- UNet, which is based on an encoder-decoder architecture. MobileNetV2 is lightweight modified U-net with a MobileNetV2 encoder.

- The U-Net architecture is built upon the Fully Convolutional Network (FCN) and modified in a way that it yields better segmentation.

- Compared to FCN-8, the two main differences are:
  (1) U-net is symmetric
  (2) the skip connections between the downsampling path and the upsampling path apply a concatenation operator instead of a sum.

- These skip connections intend to provide local information to the global information while upsampling.

- Because of its symmetry, the network has a large number of feature maps in the upsampling path, which allows to transfer information.

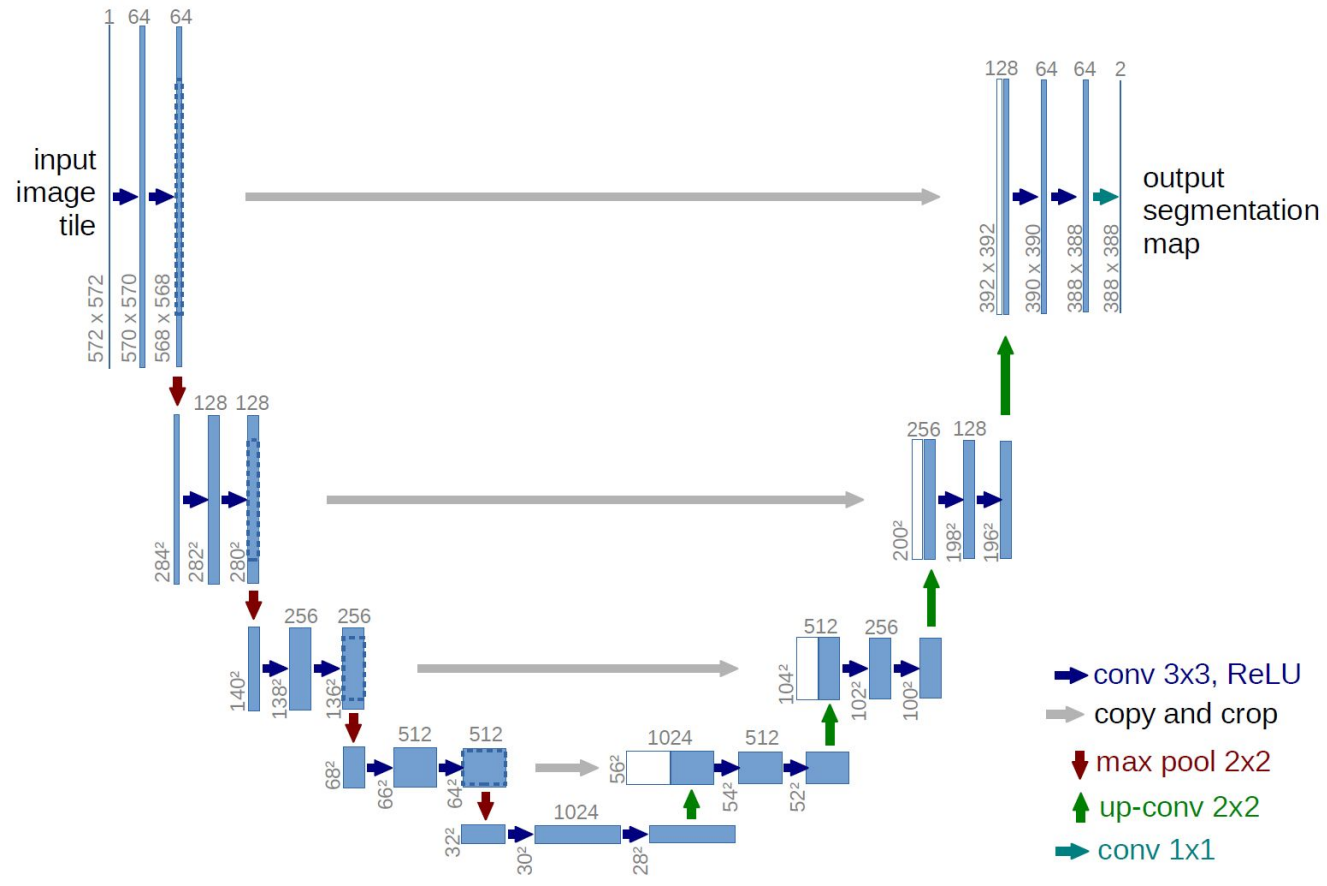
Advantage of U-net over FCN

- The main contribution of the U-Net architecture is the shortcut connections. We saw above in FCN that since we down-sample an image as part of the encoder we lost a lot of information which can't be easily recovered in the encoder part. FCN tries to address this by taking information from pooling layers before the final feature layer.

- U-Net proposes a new approach to solve this information loss problem. It proposes to send information to every up sampling layer in decoder from the corresponding down sampling layer in the encoder as can be seen in the figure above thus capturing finer information whilst also keeping the computation low.

- Since the layers at the beginning of the encoder would have more information they would bolster the up sampling operation of decoder by providing fine details corresponding to the input images thus improving the results a lot.

U-Net Architecture



Status and Future Work :

I implemented Unet and MobileNetV2 but couldnt continue further. This was because f i got infected with Covid-19 with my family in early january.

Proposed Modification :
Ensemble Scheme : Addition of implementation of other SOTA Algorithms like Unet++, Unet V3, DeepLabv3 , DeepLabv2 etc so that they all could be trained and several models are executed simultaneously and their outputs are fused together to derive the final prediction.
Such a design can reduce incorrect predictions of individual models and produce more precise segmentation masks.

This method merges the predictions of multiple models by applying majority hard voting.
(Example snippet of implementation code is on next slide)

Future Work Proposed : Ensemble Scheme

```
def apply_voting_to_ensemble_predictions(predictions, savepath=None):
    """
    This method merges the predictions of multiple models by applying majority hard voting. It takes the first
    prediction and scales it down to [0..1] from [0..255] (normalisation) to avoid overflow of pixels -
        i.e. 255 + 255 = 254 for some reason idk.
    After that, the remaining predictions are also normalised. A pixel-wise addition is peformed between all predicted
    masks. The result is an image where each pixel holds a value which is the sum of all values of this pixel
    in all predictions - (pixel_i = [0, 0, 1] + pixel_j = [0, 0, 1]) = pixel_k = [0, 0, 2].

    These pixel values (the sums) are then averaged (divided by the total number of ensembles) which gives the final
    values. The prediction mask is then conditionised - all values larger than 0.5 are set to 0 (BACKGROUND because
    black in the mask) while all the rest are set to 1 (ROAD class because will be scaled to 255).
    :param savepath: a path to save the intermediate masks to
    :param predictions: a list of predicted segmentation masks. They have to be numpy arrays with the same shape (W,H,1)
    :return: an averaged prediction computed by applying 'HARD PIXEL-WISE MAJORITY VOTING' scheme to all predictions.
    """
    # initialize the ensemble prediction image and normalize it

    voted_pred = predictions[0] / 255
    if savepath:
        Image.fromarray(predictions[0]).convert('RGB').save(os.path.join(savepath, 'prediction_1.png'))
    else:
        show_img(predictions[0])
    for j, pred in enumerate(predictions[1:]):
        if savepath:
            Image.fromarray(pred).save(os.path.join(savepath, f'prediction_{j+2}.png'))
        else:
            show_img(pred)
        voted_pred = voted_pred + (pred / 255)

    averaged_pred = voted_pred / len(predictions)
    if savepath:
        Image.fromarray((voted_pred*255).astype(np.uint8)).save(os.path.join(savepath, 'voted_prediction.png'))
        Image.fromarray((averaged_pred*255).astype(np.uint8)).save(os.path.join(savepath, 'averaged_prediction.png'))
    else:
        show_img(voted_pred)
        show_img(averaged_pred)
    averaged_pred = (averaged_pred < 0.501).astype(np.uint8)

    return averaged_pred
```

Training Details of U-net Model Trained

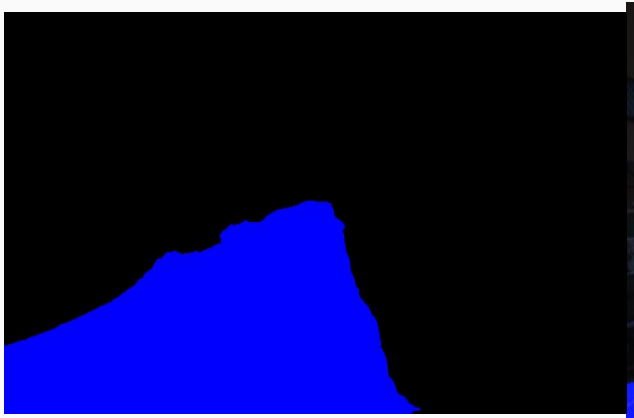
- IMG_SIZE: (128, 128)
- BATCH_SIZE: 4
- OUTPUT_CHANNELS: 2
- EPOCHS: 20
- VERSION: unet
- VAL_SPLIT: 0.2
- STEPS_PER_EPOCH: None
- PARTIAL_SAMPLING: 0.1
- LOSS: BinaryFocalLoss - Gamma(1.0)
- OPTIMIZER: Adam - LR(9.999999747378752e-05)

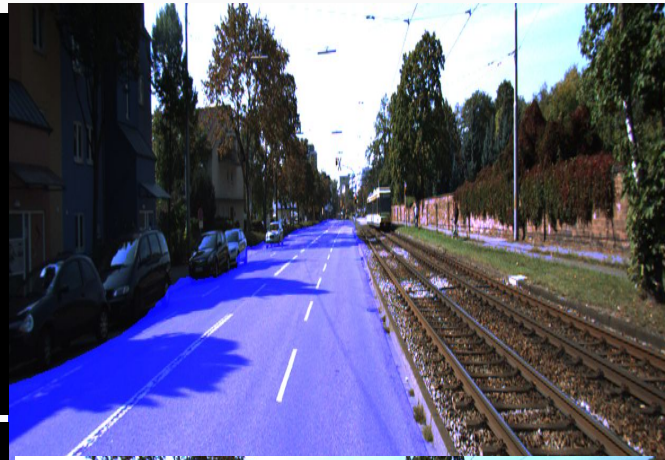
Training Results : Kitti Dataset

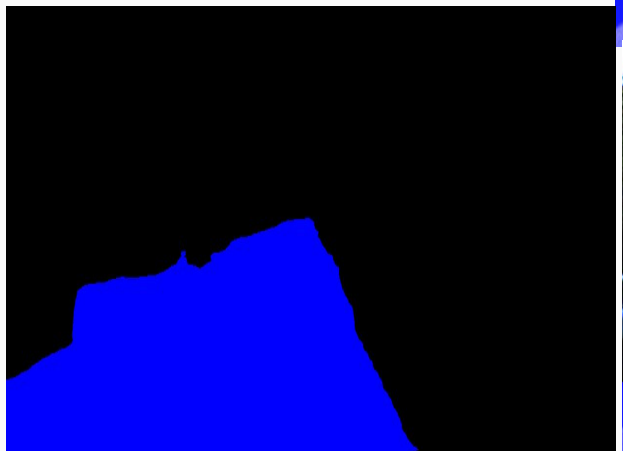
INPUT IMAGE

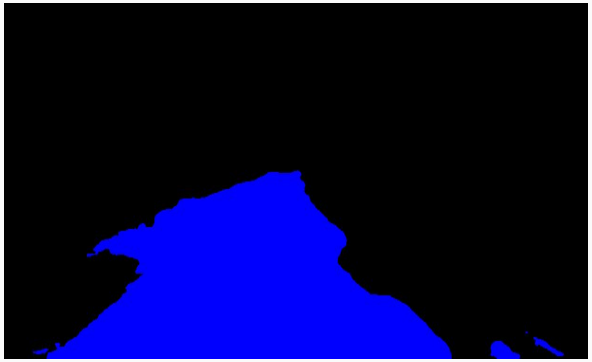
LABEL IMAGE

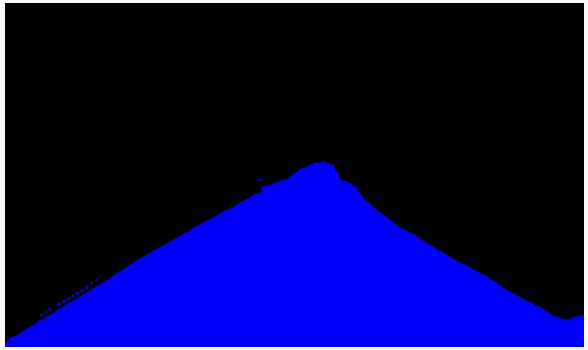
OUTPUT IMAGE

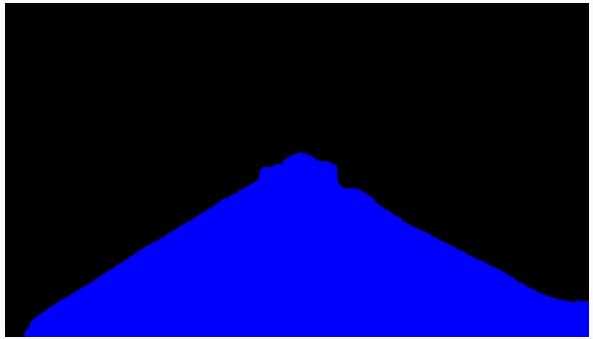


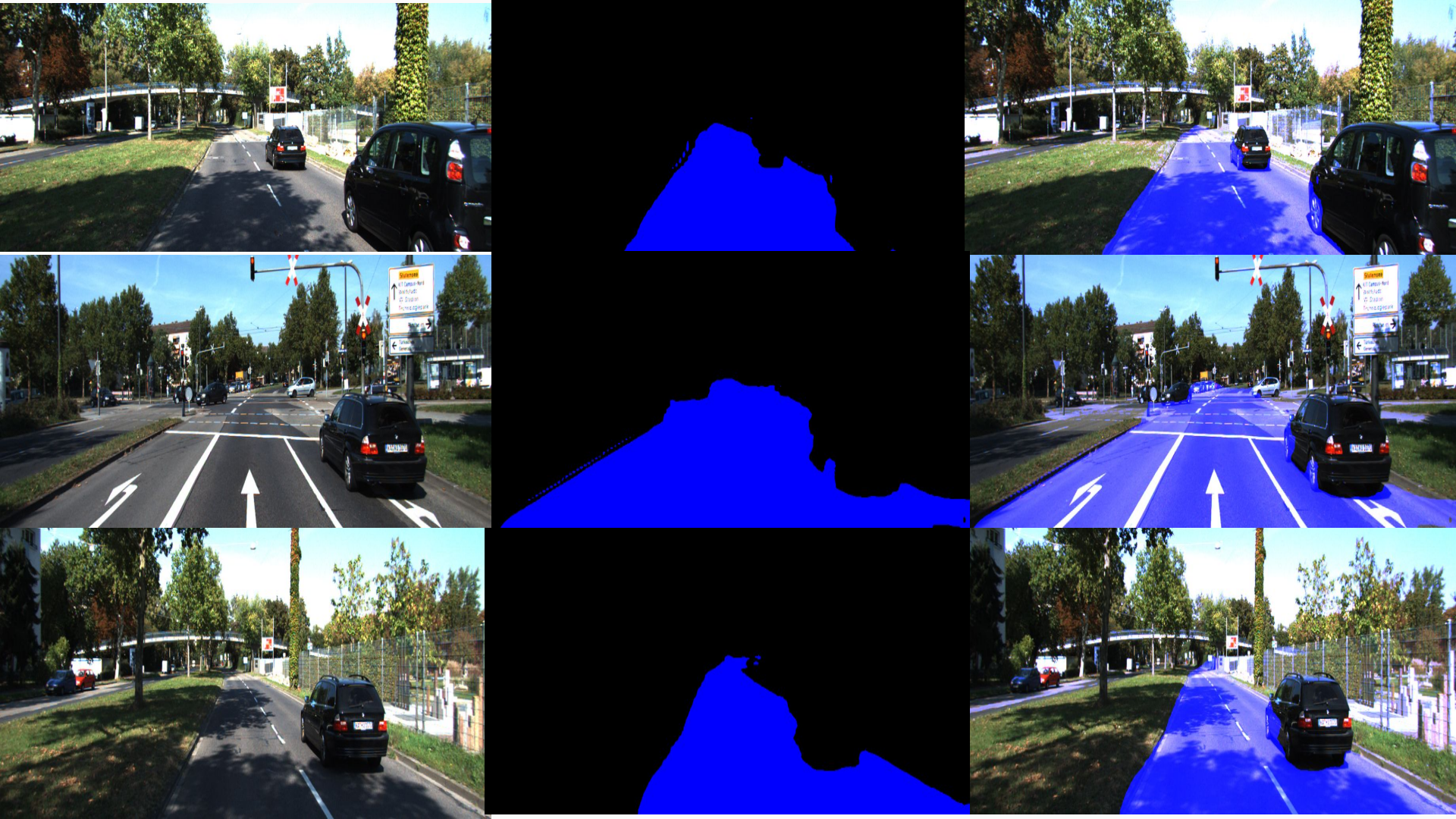


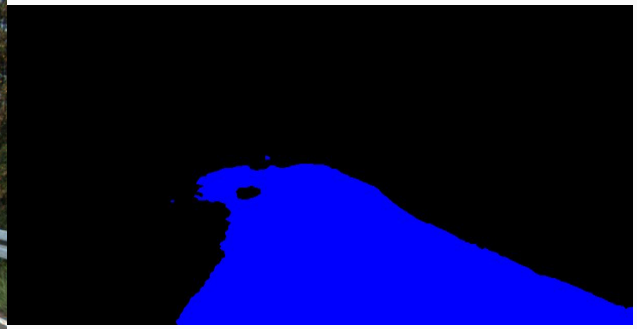


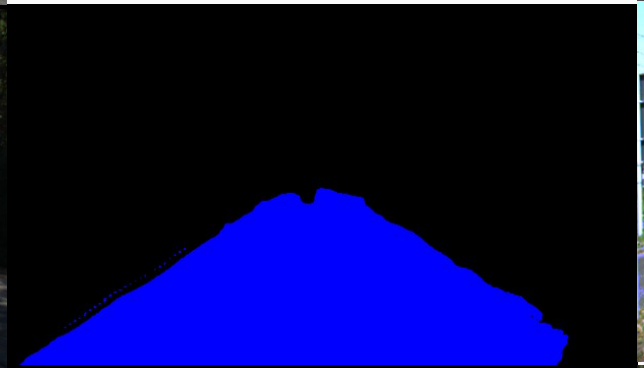



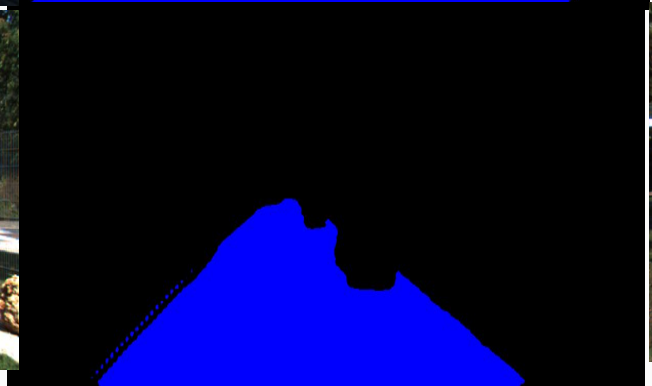



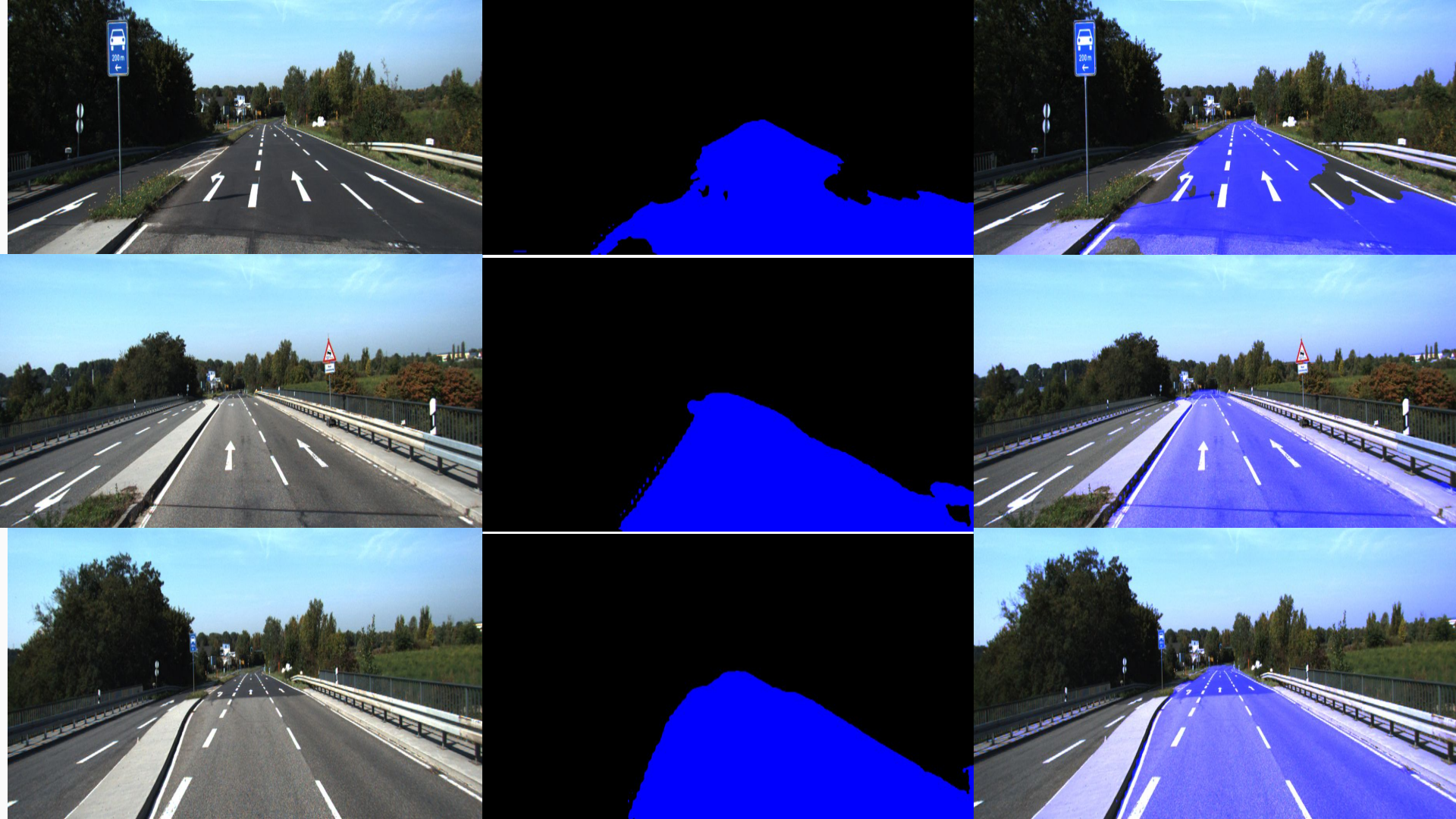



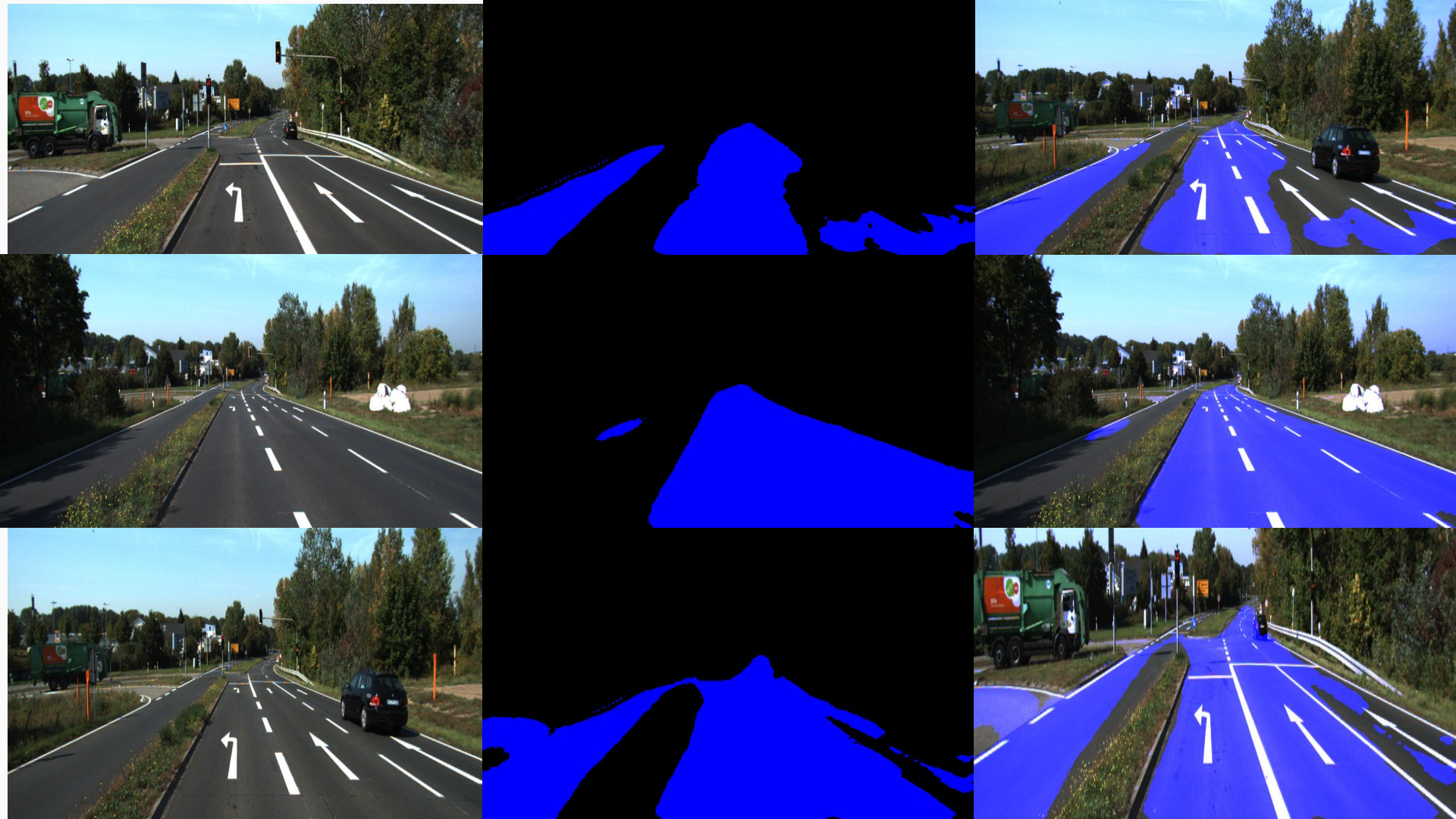



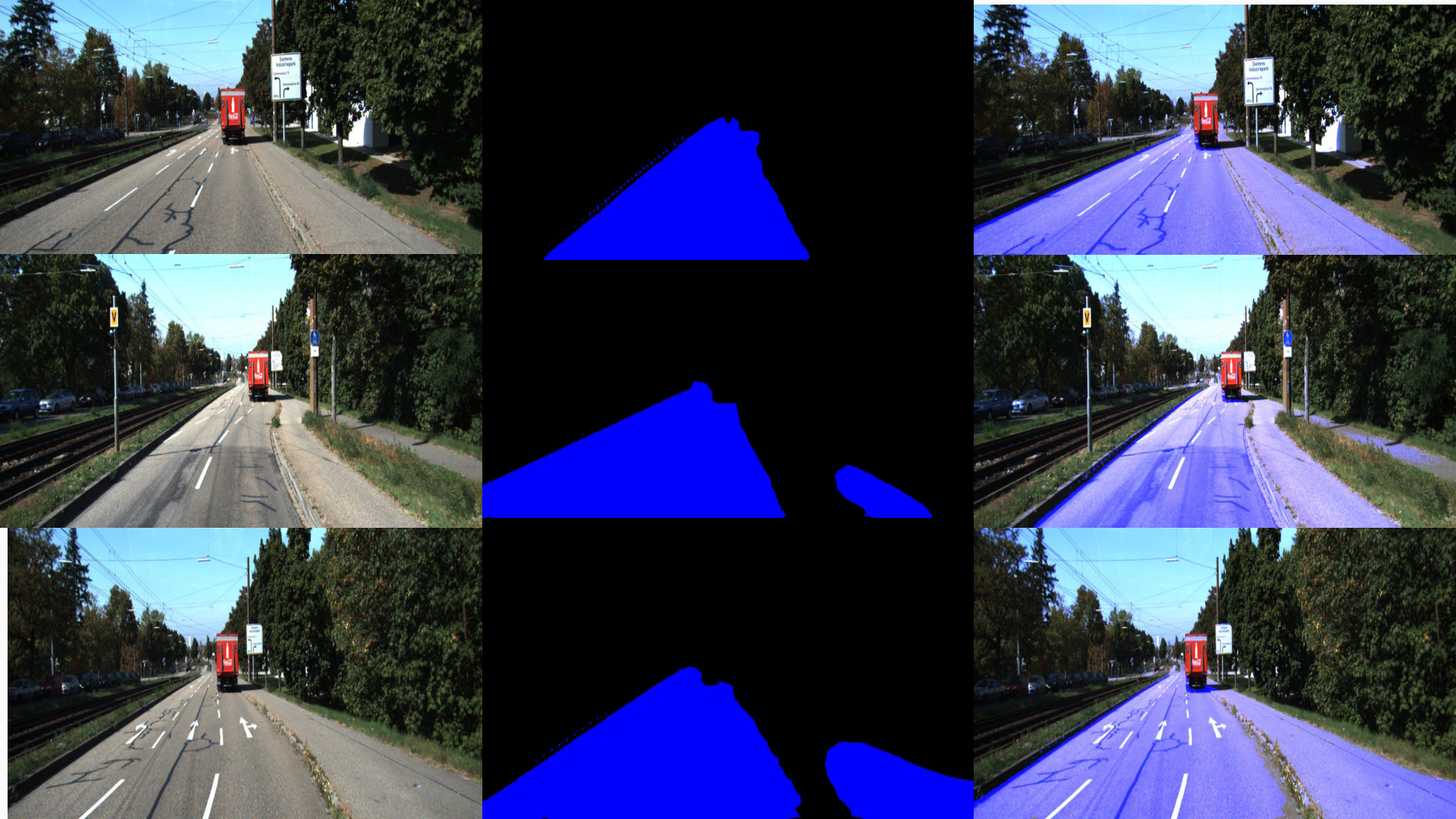



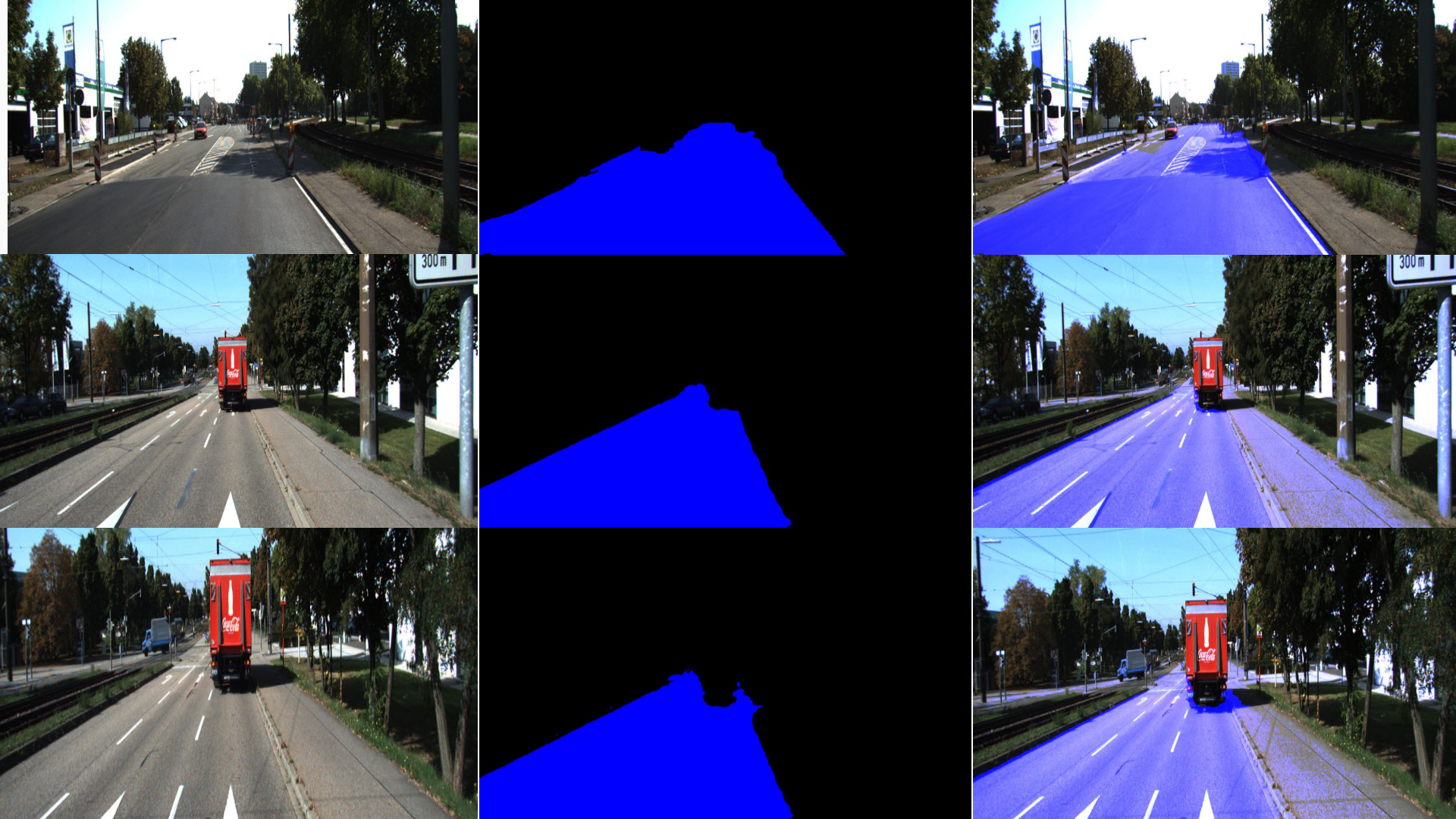



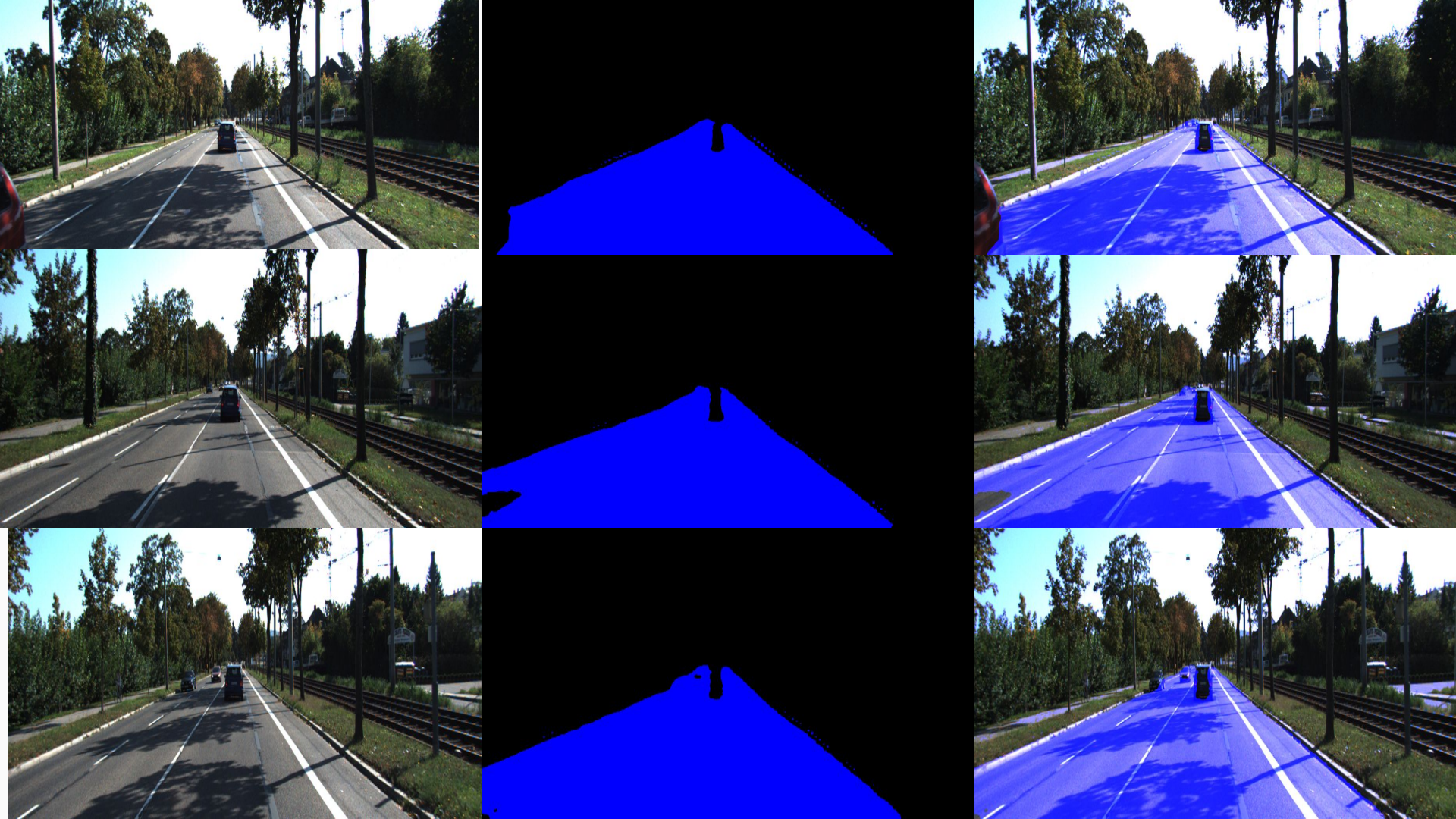



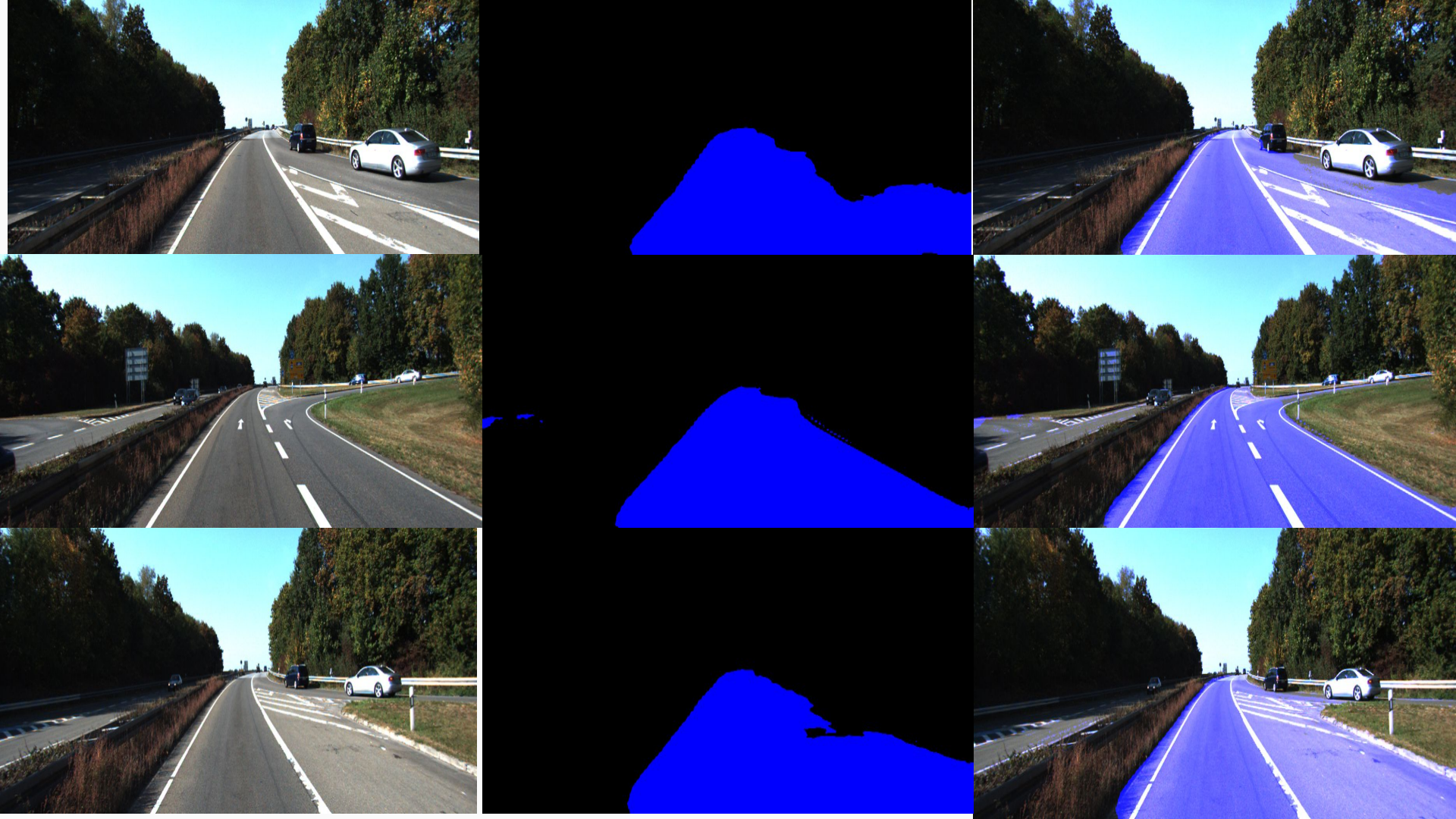



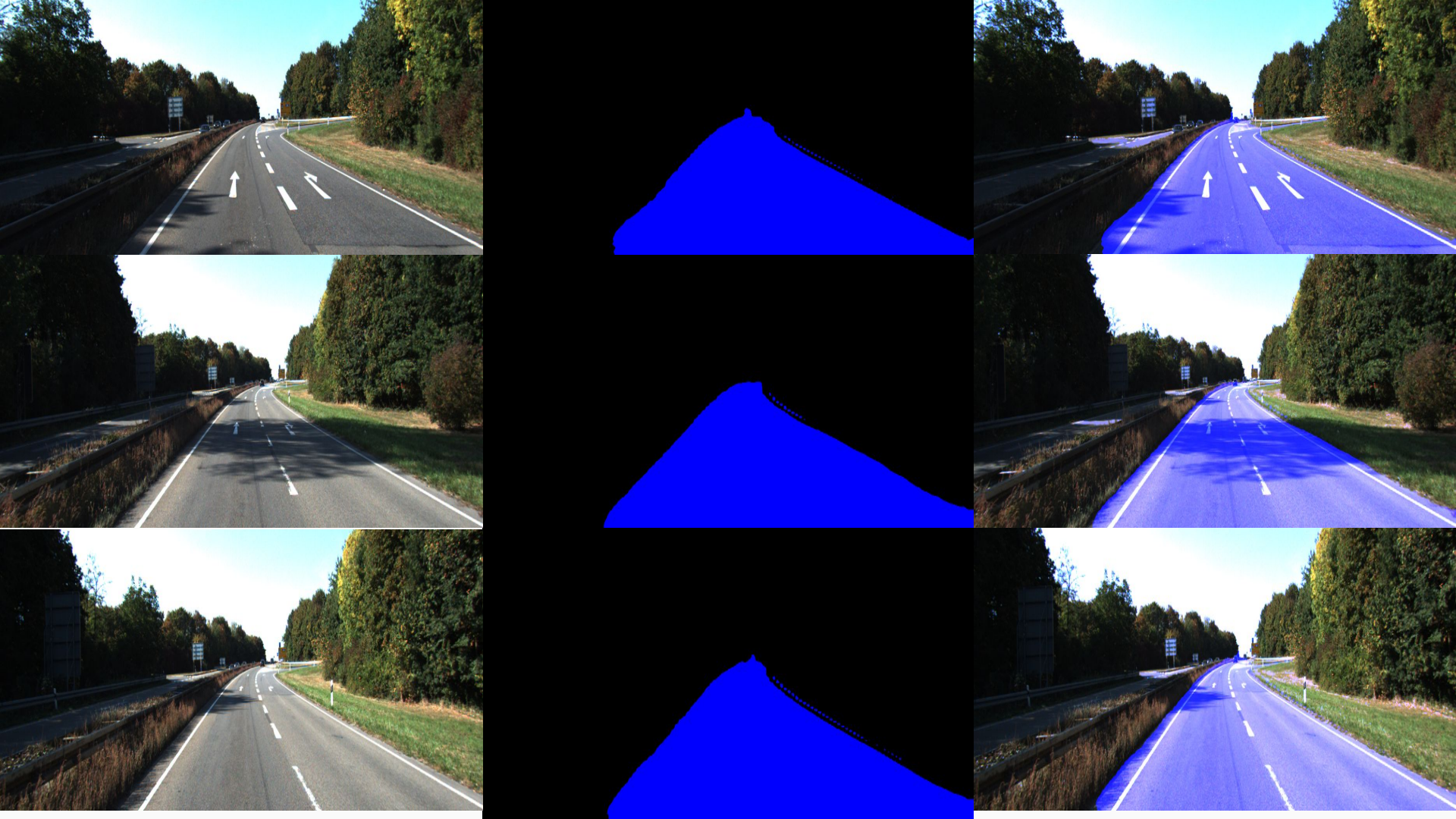


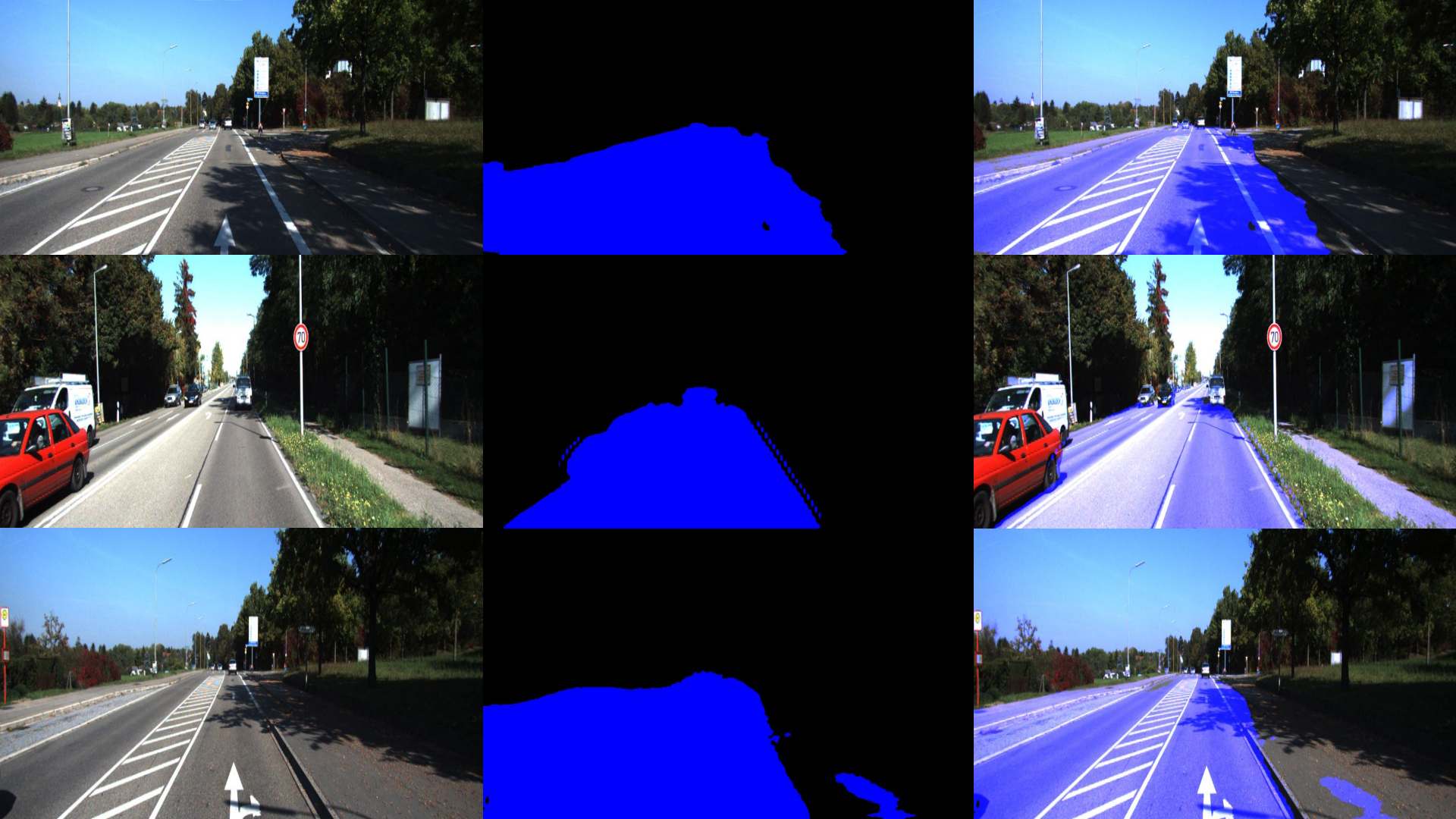


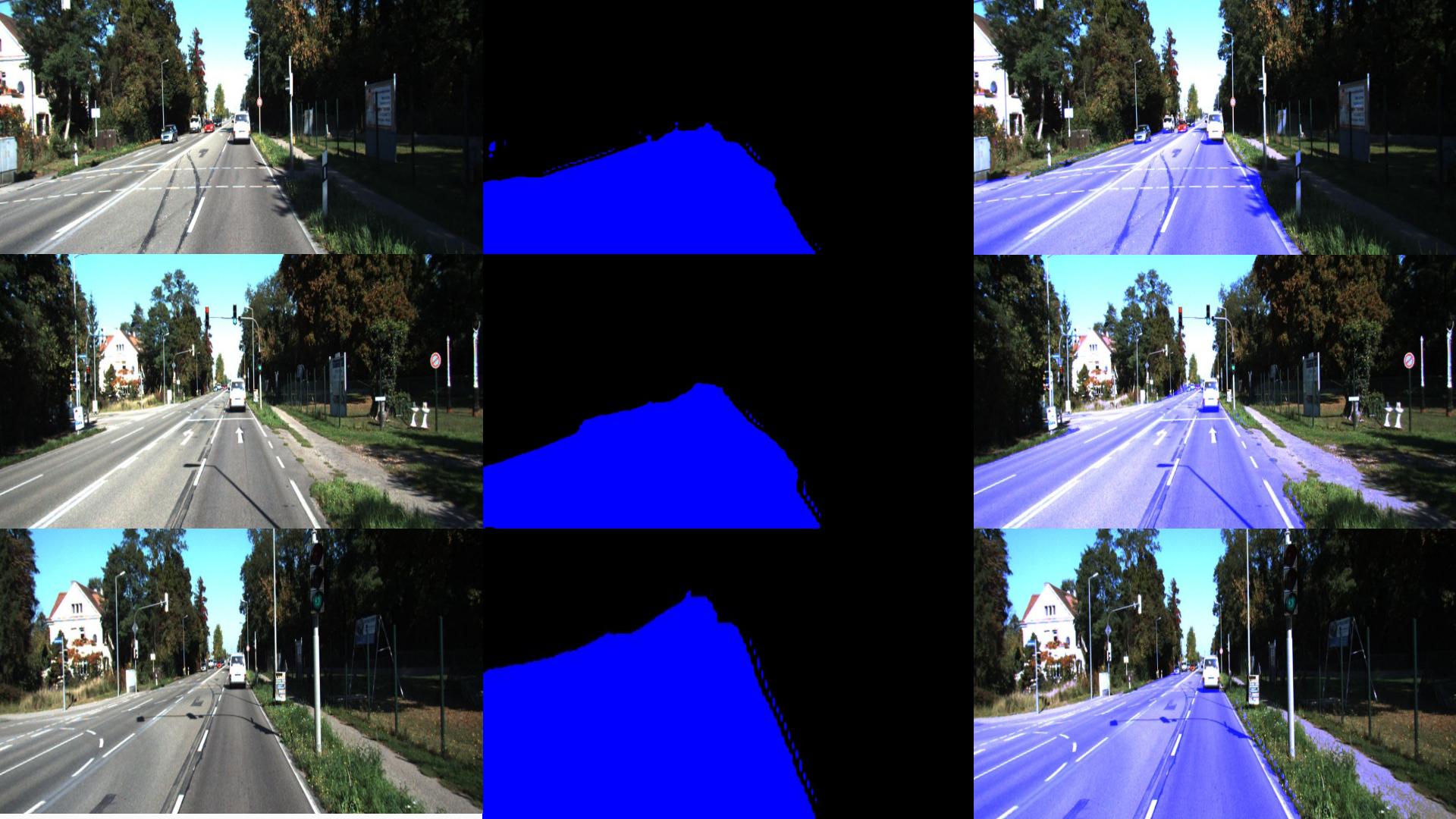


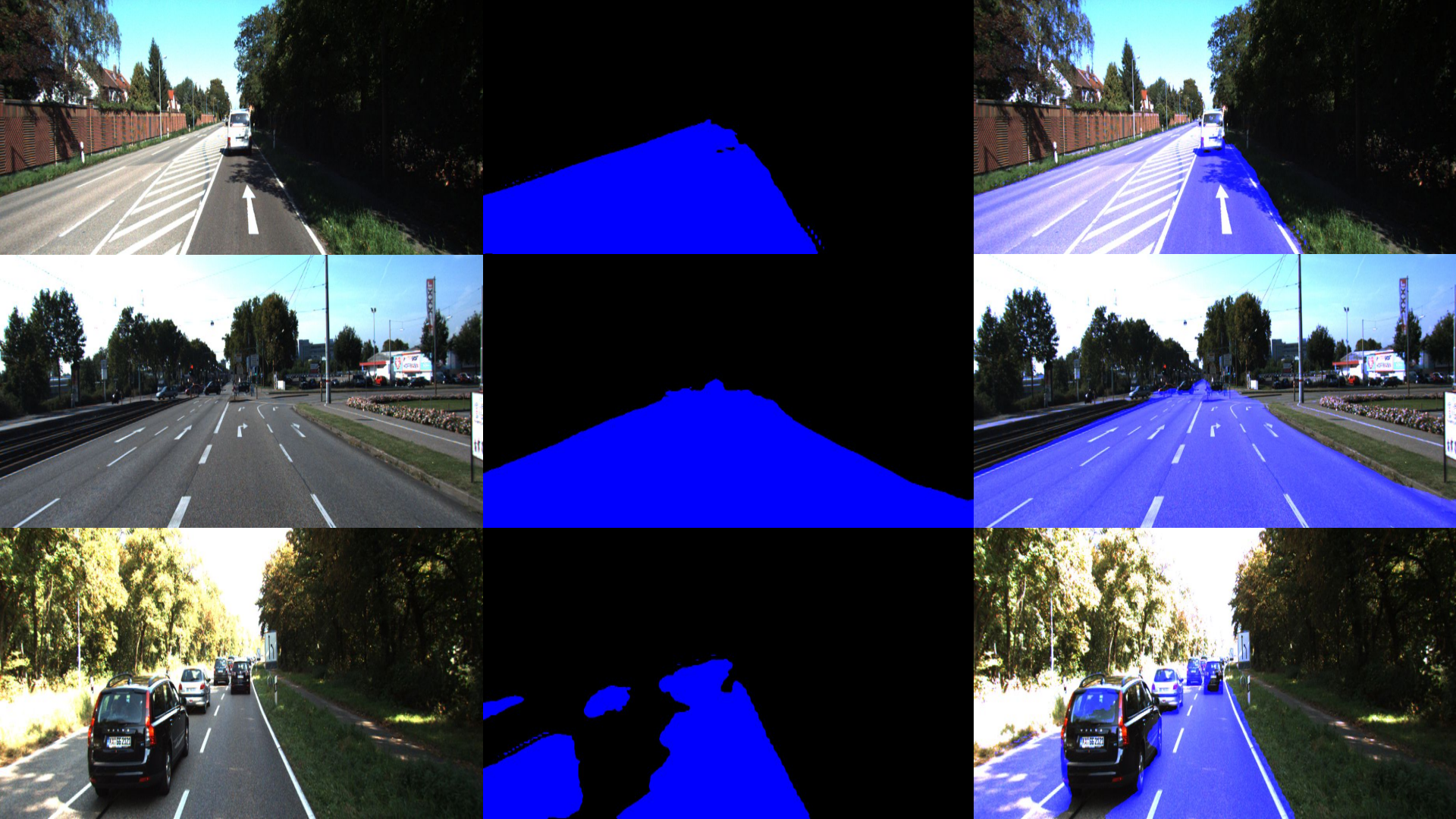








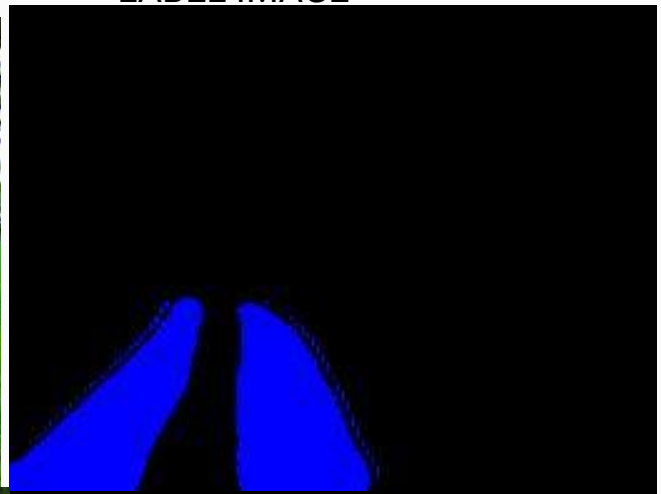




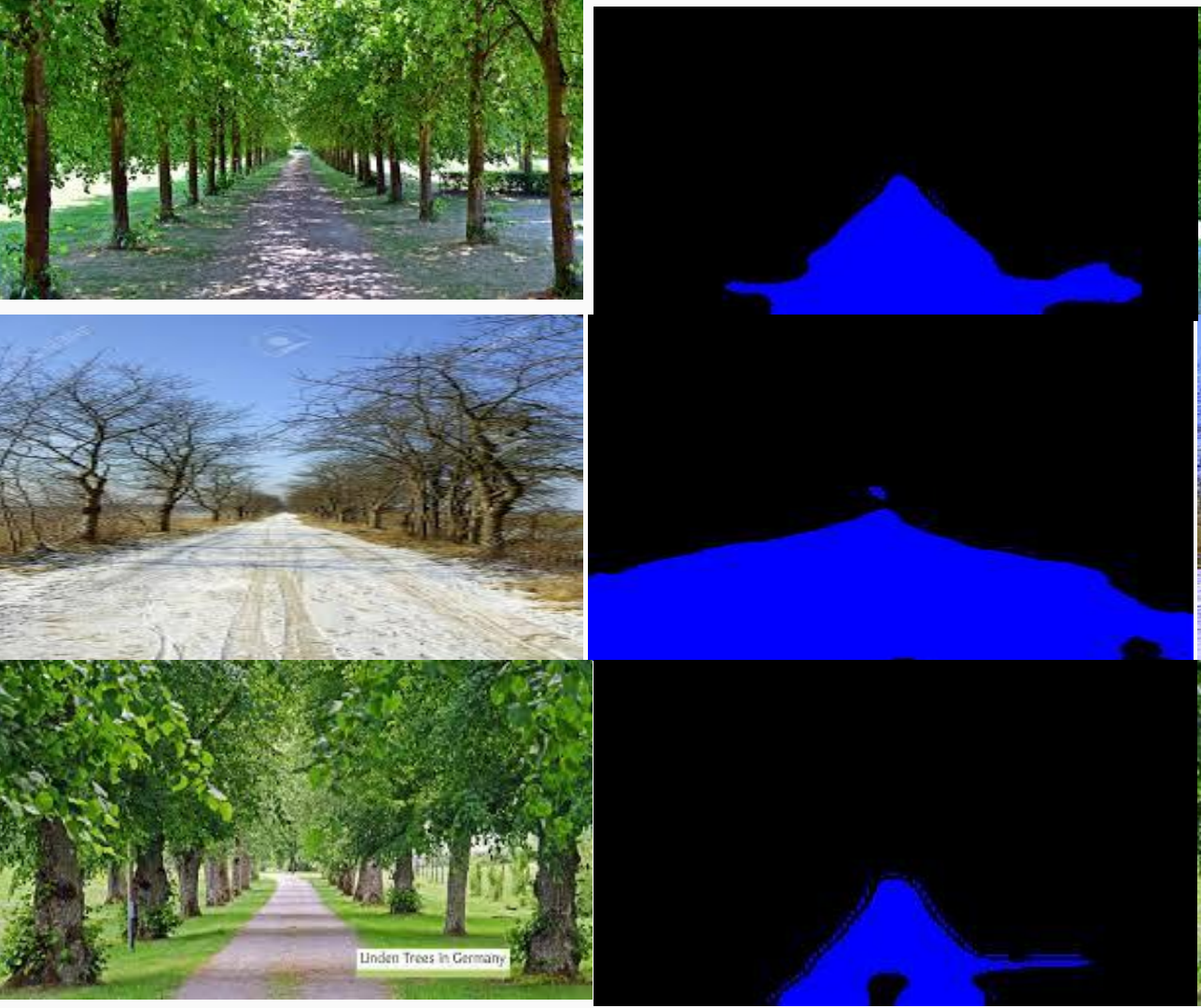
Garden Dataset :

INPUT IMAGE

LABEL IMAGE



OUTPUT IMAGE

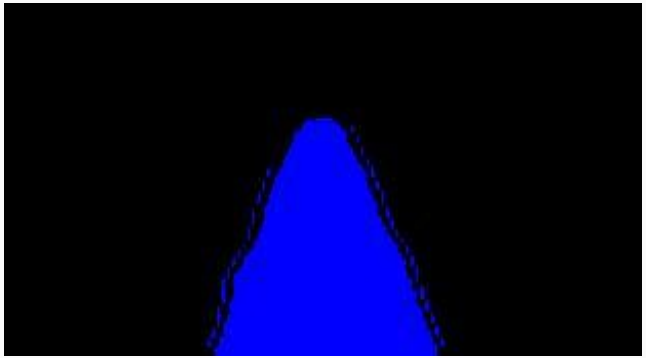

Linden Trees in Germany
Linden Trees in Germany

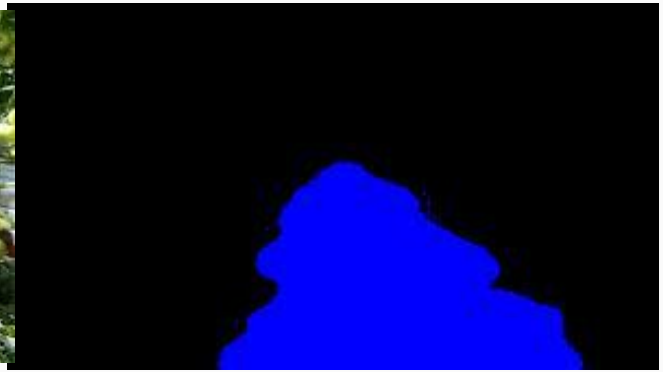
Agrofarm Dataset : 15,16 ,25,12,2

INPUT IMAGE

LABEL IMAGE

OUTPUT IMAGE










THANK YOU