Home Work 6

Name : Sai Teja Chava
Course Name : Advanced Computer Vision(CSCI 677)
USC ID : 5551847788
Instructor Name : Prof. Ram Nevatia

Steps in Code:

1. Import all the required packages.
2. Read the required parameters (epochs, batch size and path to save plots and dataset) as command line arguments.
3. Read the training data using cv2.imread(), convert it to numpy array.
4. Split the training data into training and validation sets with 199 in training and 45 in validation.
5. Read the labels using pickle.
6. Split the labels into training and testing with 199 in training and remaining in testing.
7. Read the test data and labels in similar way as in training.
8. Code the network following configuration D of VGGNet
9. Covert the FC layers to conv layers.
10. Include a deconv layer at the last after converting FC layers to conv layers and pass the parameters given in the assignment to deconv layer.
11. Define the optimizer and loss function.
12. Make sure to mask the values(-1's) when you calculate the loss.
13. Start training the network for specified number of epochs with batch size of 1.
14. Print the validation loss after every epoch so that you can monitor how the network is training.
15. Also calculate the IOU using the formulae specified in assignment.
16. Once the training is done, make a forward pass on the test data and calculate the IOU.
17. Plot the training loss and validation loss over epochs and see how the training loss and validation losses evolved over time.
18. Also plot the IOU of the validation data over number of epochs.

Note: I did not record the binary segmentation result for this experiment, but I did it for next two experiments.

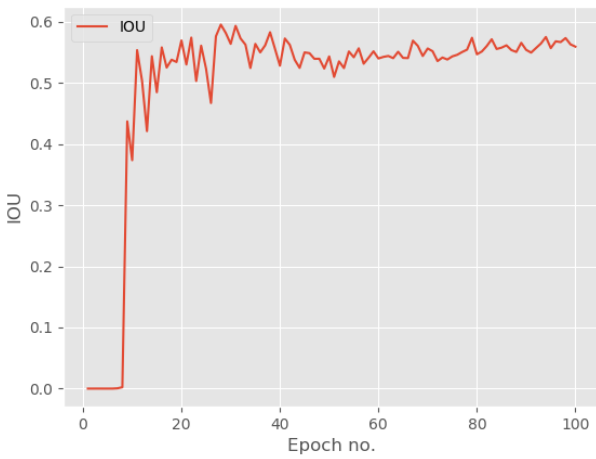
More results for test images were included in the appendix and only few were shown after discussion section in each experiment,

Results:

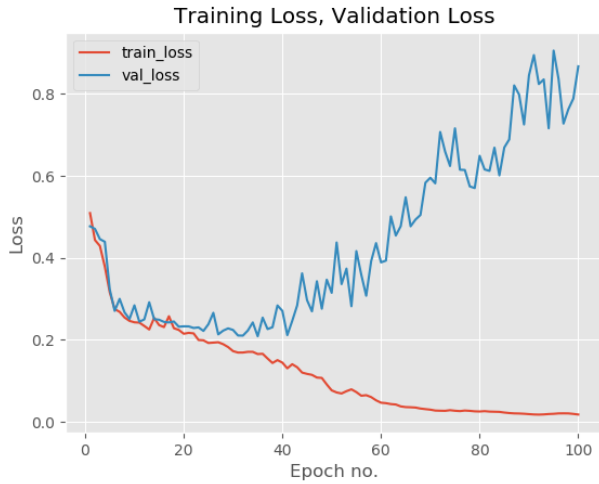
Experiment 1 (With the parameters specified in assignment but no of epochs are set to 100)

Plots:

IOU(Validation):



Losses:



Test loss : 0.86
Test IOU : 55.06

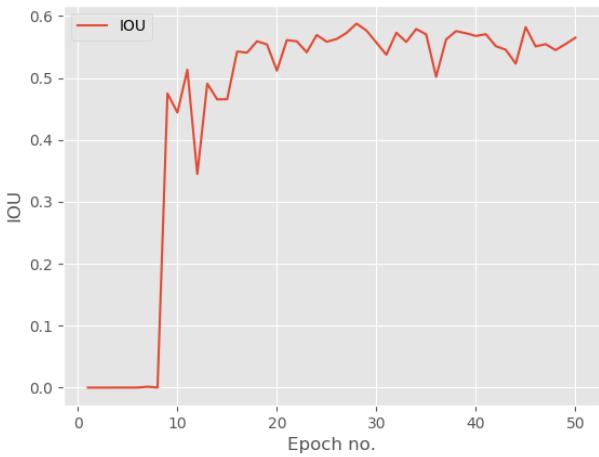
Discussion:

- As you can see from the losses plot, beyond 40th epoch the network started to overfit as training loss went down and validation loss started to increase. The gap between training loss and validation loss became wider and wider from there.
- IOU was zero for a long time like 8 epochs and after that it increases rapidly in next few epochs.
- Then the IOU fluctuates and stabilizes.
- It was really hard to find the right initializer. It took me more than 2 hours before I got a good initialization.

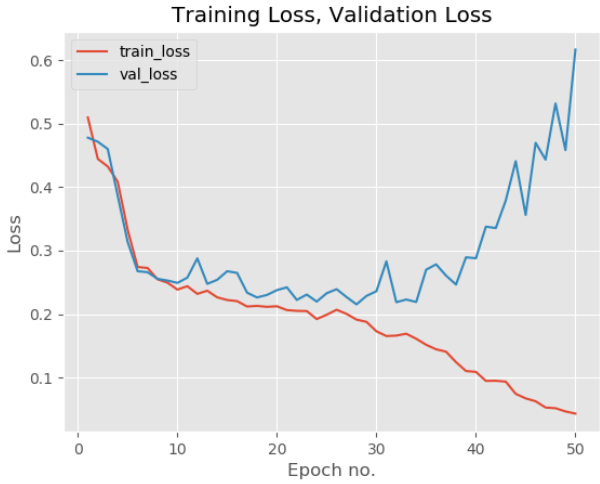
Experiment 2 (With the parameters specified in assignment but no of epochs are set to 50)

Plots:

IOU(Validation):



Losses:

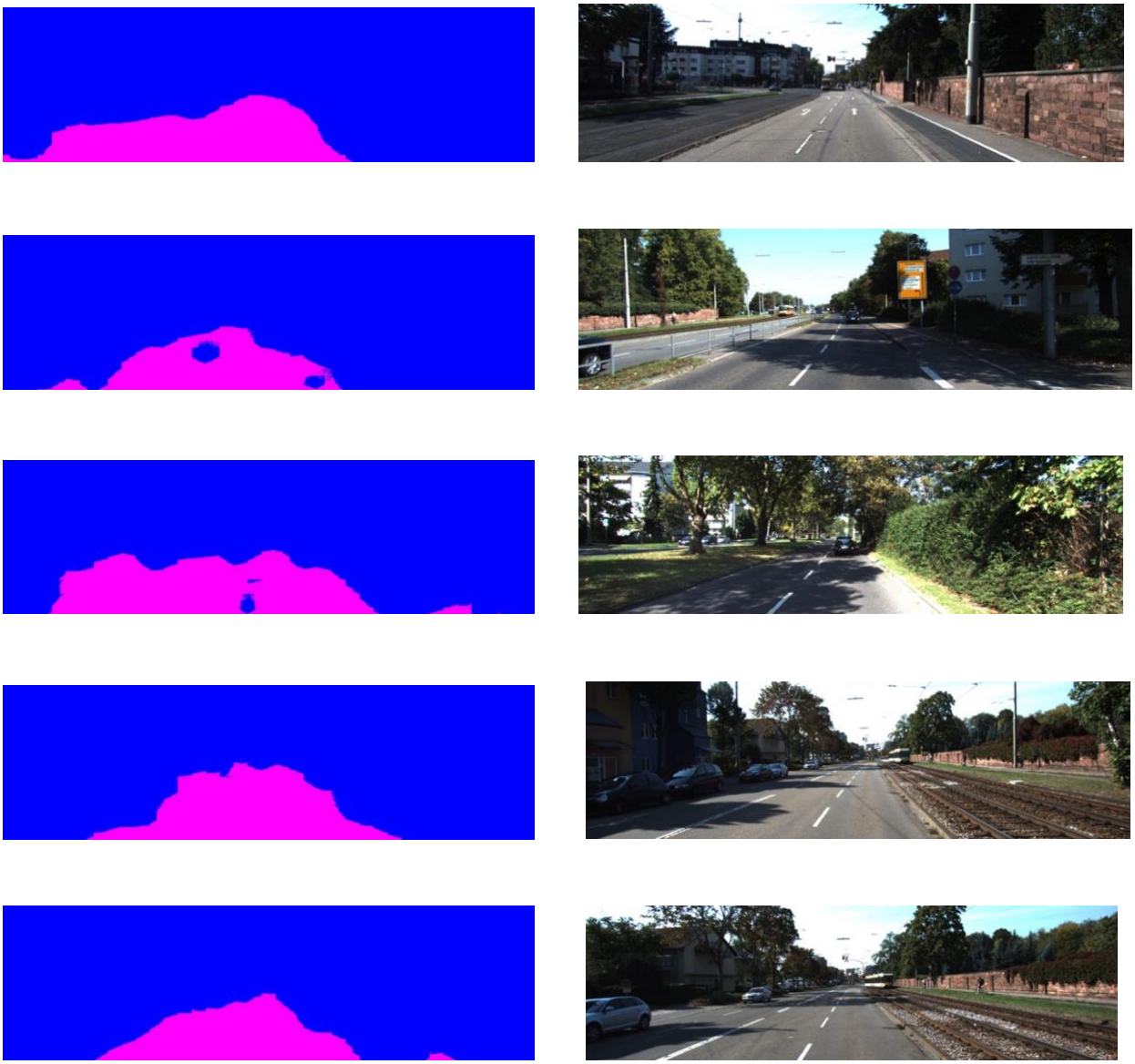


Test loss : 0.68
Test IOU : 58.06

Discussion:

- As you can see from the losses plot, beyond 30th epoch the network started to overfit as training loss went down and validation loss started to increase.
- IOU was zero for a long time like 7-8 epochs and after that it increases rapidly in next few epochs.
- Then the IOU fluctuates and stabilizes.
- It was really hard to find the right initializer. It took me more than 2 hours before I got a good initialization.

Few images results are as follows:

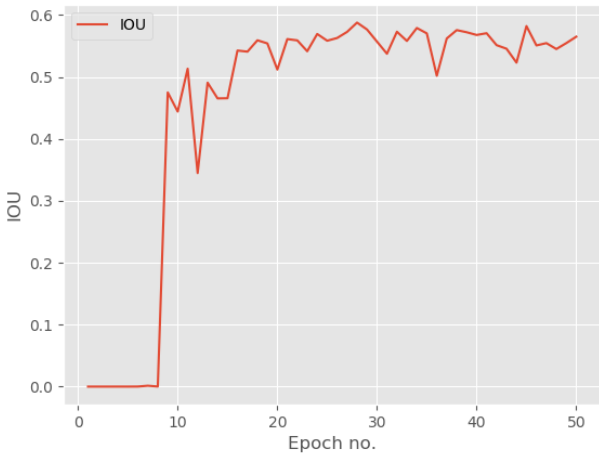


Experiment 3 (With RMSProp Optimizer, Removed ReLu before deconv layer and no of epochs are set to 50)

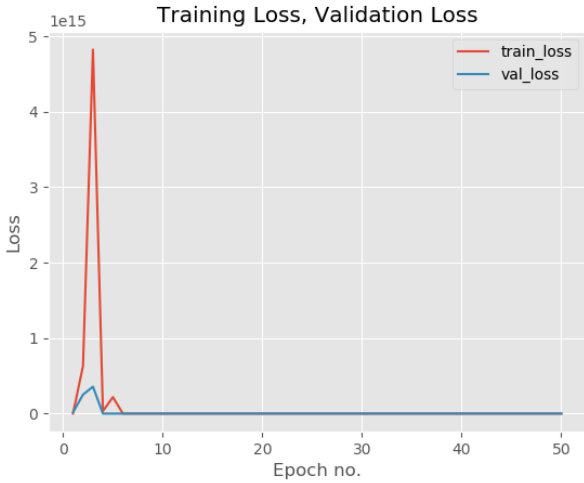
Other parameters remain unchanged.

Plots:

IOU(Validation):



Losses:

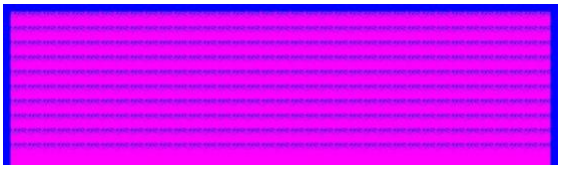


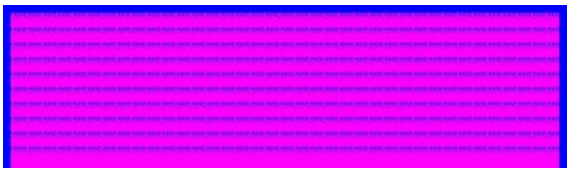
Test loss: 1.243
Test IOU: 0.183

Discussion:

- As you can see from the losses plot, training loss starts at zero and there is huge bump in training loss where it goes up to 5 and then comes down close to zero rapidly.
- Same thing happens with the validation loss too but here the height of the bump is much lesser than in training.
- IOU was zero for a long time like 7-8 epochs and after that it increases rapidly in next few epochs.
- Then the IOU fluctuates and stabilizes.
- It was really hard to find the right initializer. It took me more than 2 hours before I got a good initialization.
- Surprisingly the test IOU was much smaller than when observed with validation.
- By removing the ReLu before the deconv layer helped to find the initializer in very less time.
- Results were really bad in this experiment as you can see that test IOU was only around 18%

Few images results are as follows:



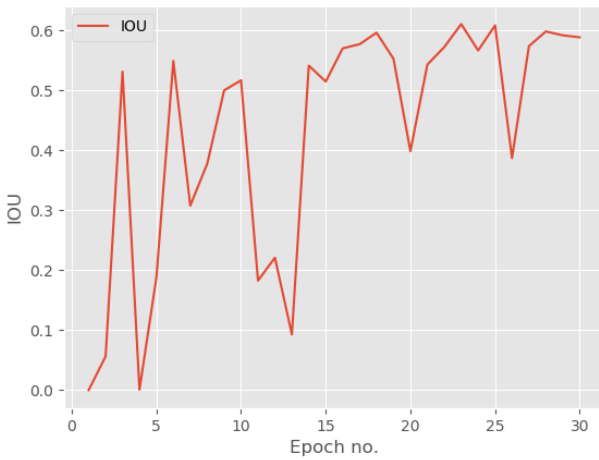


Experiment 4 (With SGDOptimizer, Removed ReLu before deconv layer and no of epochs are set to 30)

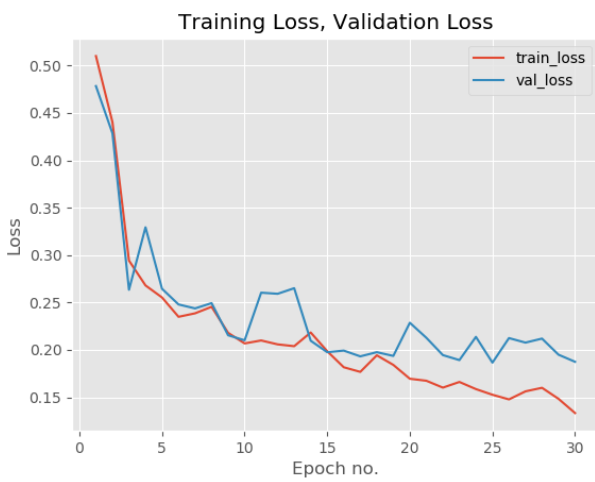
Other parameters remain unchanged.

Plots:

IOU(Validation):



Losses:



Test loss: 0.186
Test IOU: 0.587

Discussion:

- As you can see from the losses plot, training loss and validation losses stayed pretty close throughout entire training process.
- IOU was fluctuating very heavily than in any other expreiments.
- IOU and loss for test were good.
- By removing the ReLu before the deconv layer helped to find the initializer in very less time.

Few images results are as follows:

Other experiments that I tried but did not work out.

1. I tried using Adam optimizer, but it did not converge at all, all the values of predictions are really close.
2. Also tied changing the learning rate from 0.001 to 0.01 and could not get any converging results.

Summary

1. Judging the successful and failure results was not easy as many of the results were not completely good or completely bad.
2. In the 3rd experiment almost all the results were bad and I included few of them.

Code :

```
import tensorflow as tf
import os
import cv2
import argparse
import random
import pickle
import matplotlib
matplotlib.use("Agg")
from imutils import paths
import numpy as np
from tensorflow.contrib.layers import flatten
from sklearn.metrics import confusion_matrix
from sklearn.utils import shuffle
from keras.preprocessing.image import img_to_array
import matplotlib.pyplot as plt

ap = argparse.ArgumentParser()
ap.add_argument("-e","--epochs",required = True, type = str, help = "Enter the no of epochs")
ap.add_argument("-b","--batch_size",required = True, type = str, help = "Enter the batch size")
ap.add_argument("-d","--dataset", required = True, type = str, help = "Enter the path to dataset")
ap.add_argument("-p","--plot",required = True, type = str, help = "Enter the path to the plot of
training and validation losses")
ap.add_argument("-i","--IOU",required = True, type = str, help = "Enter the path to the plot IOU")
args = vars(ap.parse_args())


def randomize(dataset, labels):
    permutation = np.random.permutation(labels.shape[0])
    shuffled_dataset = dataset[permutation, :, :]
```

```
    shuffled_labels = labels[permutation]
    return shuffled_dataset, shuffled_labels

# tf.truncated_normal outputs random values from a truncated normal distribution.
# Genereated values follow a normal distribution with specified mean and standard deviation.
# Initialize weights variable with random values.
def init_weight(shape):
    #w = tf.truncated_normal(shape=shape, mean = 0, stddev = 0.1)
    initializer = tf.contrib.layers.xavier_initializer()
    return tf.Variable(initializer(shape))

# Initialize the bias variable with zeros.
def init_bias(shape):
    b = tf.zeros(shape)
    return tf.Variable(b)

def VGGNet(x):
    #Conv1
    conv1_W = init_weight((3,3,3,64))
    conv1_b = init_bias(64)
    conv1   = tf.nn.conv2d(x, conv1_W, strides=[1, 1, 1, 1], padding='SAME') + conv1_b
    conv1 = tf.nn.relu(conv1)

    #Conv2
    conv2_W = init_weight((3,3,64,64))
    conv2_b = init_bias(64)
    conv2   = tf.nn.conv2d(conv1, conv2_W, strides=[1, 1, 1, 1], padding='SAME') + conv2_b
    conv2 = tf.nn.relu(conv2)

    #Max pool
    conv2 = tf.nn.max_pool(conv2, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')

    #Conv3
    conv3_W = init_weight((3,3,64,128))
    conv3_b = init_bias(128)
    conv3   = tf.nn.conv2d(conv2, conv3_W, strides=[1, 1, 1, 1], padding='SAME') + conv3_b
    conv3 = tf.nn.relu(conv3)

    #Conv4
    conv4_W = init_weight((3,3,128,128))
    conv4_b = init_bias(128)
    conv4   = tf.nn.conv2d(conv3, conv4_W, strides=[1, 1, 1, 1], padding='SAME') + conv4_b
    conv4 = tf.nn.relu(conv4)

    #Max pool
    conv4 = tf.nn.max_pool(conv4, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')
```

```
#Conv5
conv5_W = init_weight((3,3,128,256))
conv5_b = init_bias(256)
conv5   = tf.nn.conv2d(conv4, conv5_W, strides=[1, 1, 1, 1], padding='SAME') + conv5_b
conv5 = tf.nn.relu(conv5)

#Conv6
conv6_W = init_weight((3,3,256,256))
conv6_b = init_bias(256)
conv6   = tf.nn.conv2d(conv5, conv6_W, strides=[1, 1, 1, 1], padding='SAME') + conv6_b
conv6 = tf.nn.relu(conv6)

#Conv7
conv7_W = init_weight((3,3,256,256))
conv7_b = init_bias(256)
conv7   = tf.nn.conv2d(conv6, conv7_W, strides=[1, 1, 1, 1], padding='SAME') + conv7_b
conv7 = tf.nn.relu(conv7)

#Max pool
conv7 = tf.nn.max_pool(conv7, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')

#Conv8
conv8_W = init_weight((3,3,256,512))
conv8_b = init_bias(512)
conv8   = tf.nn.conv2d(conv7, conv8_W, strides=[1, 1, 1, 1], padding='SAME') + conv8_b
conv8 = tf.nn.relu(conv8)

#Conv9
conv9_W = init_weight((3,3,512,512))
conv9_b = init_bias(512)
conv9   = tf.nn.conv2d(conv8, conv9_W, strides=[1, 1, 1, 1], padding='SAME') + conv9_b
conv9 = tf.nn.relu(conv9)

#Conv10
conv10_W = init_weight((3,3,512,512))
conv10_b = init_bias(512)
conv10   = tf.nn.conv2d(conv9, conv10_W, strides=[1, 1, 1, 1], padding='SAME') + conv10_b
conv10 = tf.nn.relu(conv10)

#Max pool
conv10 = tf.nn.max_pool(conv10, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')

#Conv11
conv11_W = init_weight((3,3,512,512))
conv11_b = init_bias(512)
```

```
    conv11   = tf.nn.conv2d(conv10, conv11_W, strides=[1, 1, 1, 1], padding='SAME') + conv11_b
    conv11 = tf.nn.relu(conv11)

    #Conv12
    conv12_W = init_weight((3,3,512,512))
    conv12_b = init_bias(512)
    conv12   = tf.nn.conv2d(conv11, conv12_W, strides=[1, 1, 1, 1], padding='SAME') + conv12_b
    conv12 = tf.nn.relu(conv12)

    #Conv13
    conv13_W = init_weight((3,3,512,512))
    conv13_b = init_bias(512)
    conv13   = tf.nn.conv2d(conv12, conv13_W, strides=[1, 1, 1, 1], padding='SAME') + conv13_b
    conv13 = tf.nn.relu(conv13)

    #Max pool
    conv13 = tf.nn.max_pool(conv13, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')

    print("After conv 13 is : "+str(conv13.shape))
    #Replacing FC1 by conv
    conv14_W = init_weight((7,7,512,4096))
    conv14_b = init_bias(4096)
    conv14   = tf.nn.conv2d(conv13, conv14_W, strides=[1, 1, 1, 1], padding='SAME') + conv14_b
    conv14 = tf.nn.relu(conv14)

    print("After FC1 is : "+str(conv14.shape))
    #Replacing FC2 by conv
    conv15_W = init_weight((1,1,4096,4096))
    conv15_b = init_bias(4096)
    conv15   = tf.nn.conv2d(conv14, conv15_W, strides=[1, 1, 1, 1], padding='SAME') + conv15_b
    conv15 = tf.nn.relu(conv15)

    print("After FC2 is : "+str(conv15.shape))
    #Replacing FC3 by conv
    conv16_W = init_weight((1,1,4096,1))
    conv16_b = init_bias(1)
    conv16   = tf.nn.conv2d(conv15, conv16_W, strides=[1, 1, 1, 1], padding='SAME') + conv16_b
    #conv16 = tf.nn.relu(conv16)
    print("After FC3 is : "+str(conv16.shape))
    #Deconv layer
    deconv = tf.layers.conv2d_transpose(conv16, filters=1, strides=32,
kernel_size=64,padding='SAME')

    print("The shape of deconv is : "+str(deconv.shape))
    logits = deconv
```

```
    return logits

print("[INFO] Loading images")

data = []
labels = []

# grab the image paths and randomly shuffle them
imagePaths = list(paths.list_images(args["dataset"]+"/image/train"))

# loop over the input images
for imagePath in imagePaths:
    # load the image, pre-process it, and store it in the data list
    image = cv2.imread(imagePath)
    image = img_to_array(image)
    data.append(image)

# scale the raw pixel intensities to the range [0, 1]
data = np.array(data, dtype="float") # / 255.0

train_data = data[:199,:]
validation_data = data[199:,:]

cv2.imwrite("train_image.jpg",train_data[0])
cv2.imwrite("train_image1.jpg",train_data[1])
cv2.imwrite("train_image2.jpg",train_data[2])
cv2.imwrite("train_image3.jpg",train_data[3])

cv2.imwrite("validation_image.jpg",validation_data[0])
cv2.imwrite("validation_image1.jpg",validation_data[1])
cv2.imwrite("validation_image2.jpg",validation_data[2])
cv2.imwrite("validation_image3.jpg",validation_data[3])

directory_name = args["dataset"]+"/label/train"
for f in os.listdir(directory_name):
    label = pickle.load(open(directory_name + "/" + f,"rb"),encoding="bytes")
    labels.append(label)

labels = np.array(labels)

train_labels = labels[:199,:]
print("Shape of train labels is : "+str(train_labels.shape))
validation_labels = labels[199:,:]

train_data, train_labels = randomize(train_data,train_labels)
```

```
print("Train ")
print("max is : "+str(np.amax(train_labels[0])))
print("min is : "+str(np.amin(train_labels[0])))

print("Validation")
print("max is : "+str(np.amax(validation_labels[0])))
print("min is : "+str(np.amin(validation_labels[0])))

test_data = []
test_labels = []

# grab the image paths and randomly shuffle them
imagePaths = list(paths.list_images(args["dataset"]+"/image/test"))

# loop over the input images
for imagePath in imagePaths:
    # load the image, pre-process it, and store it in the data list
    image = cv2.imread(imagePath)
    image = img_to_array(image)
    test_data.append(image)

# scale the raw pixel intensities to the range [0, 1]
test_data = np.array(test_data, dtype="float") # / 255.0

cv2.imwrite("test_image.jpg",test_data[0])
cv2.imwrite("test_image1.jpg",test_data[1])
cv2.imwrite("test_image2.jpg",test_data[2])
cv2.imwrite("test_image3.jpg",test_data[3])

directory_name = args["dataset"]+"/label/test"
for f in os.listdir(directory_name):
    label = pickle.load(open(directory_name + "/" + f,"rb"),encoding="bytes")
    test_labels.append(label)

test_labels = np.array(test_labels)

print("Test")
print("max is : "+str(np.amax(test_labels[0])))
print("min is : "+str(np.amin(test_labels[0])))

print("\nSome meta information is as follows \n")
print("train data shape is : "+str(train_data.shape))
print("train labels shape is : "+str(train_labels.shape))
print("validation data shape is : "+str(validation_data.shape))
print("validation labels shape is : "+str(validation_labels.shape))
print("test data shape is : "+str(test_data.shape))
```

```
print("test labels shape is : "+str(test_labels.shape))

EPOCHS = int(args["epochs"])
BATCH_SIZE = int(args["batch_size"])

x = tf.placeholder(tf.float32, shape = [None,352,1216,3])
y = tf.placeholder(tf.float32,shape = [None,352,1216])

y_reshape = tf.reshape(y,[-1])

logits = VGGNet(x)
#output_reshape = tf.reshape(logits,[-1])
#output = tf.nn.sigmoid(logits)
mask = y>=0
cross_entropy =
tf.nn.sigmoid_cross_entropy_with_logits(logits=tf.boolean_mask(logits[:,:,:,0],mask),
labels=tf.boolean_mask(y,mask))
#cross_entropy = tf.nn.sigmoid_cross_entropy_with_logits(logits=logits[:,:,:,0], labels=y)
#y_aftermask = tf.boolean_mask(y,mask)

loss_operation = tf.reduce_mean(cross_entropy)
optimizer = tf.train.MomentumOptimizer(learning_rate = 0.001,momentum=0.99)
training_operation = optimizer.minimize(loss_operation)

saver = tf.train.Saver()

def evaluate(X_data, y_data):
    total_loss = 0
    num_examples = len(X_data)
    total_accuracy = 0
    sess = tf.get_default_session()
    tp = 0
    tn = 0
    fp = 0
    fn = 0
    for offset in range(0, num_examples, BATCH_SIZE):
        batch_x, batch_y = X_data[offset:offset+BATCH_SIZE],
y_data[offset:offset+BATCH_SIZE]
        loss_val = sess.run(loss_operation, feed_dict={x: batch_x, y: batch_y})
        total_loss+=loss_val
        pred = sess.run(logits, feed_dict={x:batch_x})
        min_val = np.amin(pred)
        max_val = np.amax(pred)
        print("Min val is : "+str(min_val))
        print("Max val is : "+str(max_val))
        pred[pred>0]=1
```

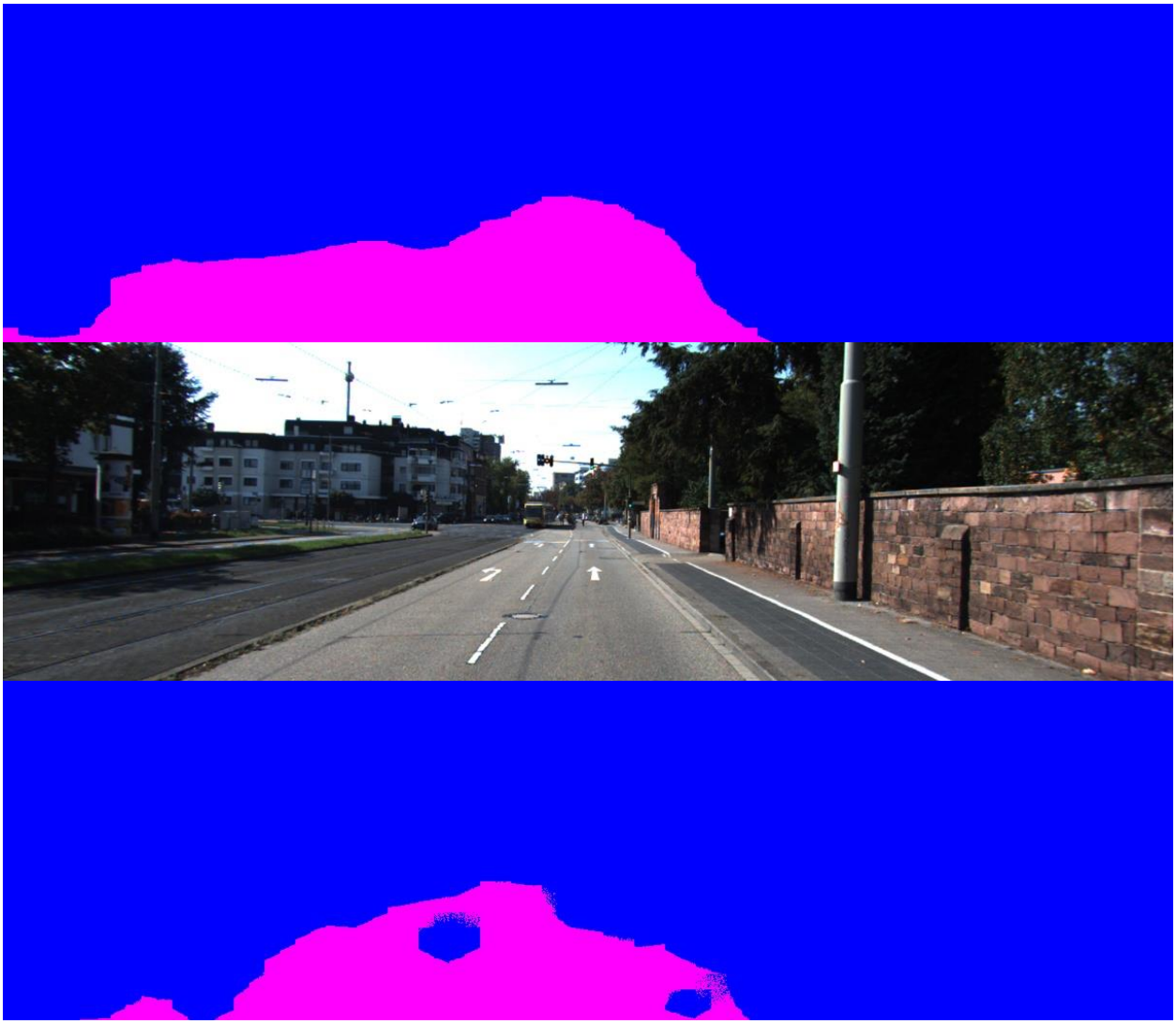
```
        pred[pred<=0]=0
        TP = np.sum(np.logical_and(pred[:,:,:,0] == 1, batch_y == 1))
        TN = np.sum(np.logical_and(pred[:,:,:,0] == 0, batch_y == 0))
        FP = np.sum(np.logical_and(pred[:,:,:,0] == 1, batch_y == 0))
        FN = np.sum(np.logical_and(pred[:,:,:,0] == 0, batch_y == 1))
        tp+=TP
        tn+=TN
        fp+=FP
        fn+=FN
    no_of_batches = num_examples/BATCH_SIZE
    total_loss = total_loss/no_of_batches
    IOU = (float(tp))/(tp+fp+fn)
    print("TP is : "+str(tp))
    print("TN is : "+str(tn))
    print("FP is : "+str(fp))
    print("FN is : "+str(fn))
    return total_loss,IOU

def evaluate1(X_data, y_data):
    total_loss = 0
    num_examples = len(X_data)
    total_accuracy = 0
    sess = tf.get_default_session()
    tp = 0
    tn = 0
    fp = 0
    fn = 0
    file_name = "test_"
    i = 0
    for offset in range(0, num_examples, BATCH_SIZE):
        batch_x, batch_y = X_data[offset:offset+BATCH_SIZE],
y_data[offset:offset+BATCH_SIZE]
        loss_val = sess.run(loss_operation, feed_dict={x: batch_x, y: batch_y})
        total_loss+=loss_val
        f_n = file_name+str(i)+".jpg"
        f_np = file_name+str(i)+"_pred.jpg"
        print(f_n)
        cv2.imwrite(f_n,batch_x[0,:,:,:])
        print(batch_x.shape)
        pred = sess.run(logits, feed_dict={x:batch_x})
        min_val = np.amin(pred)
        max_val = np.amax(pred)
        print("Min val is : "+str(min_val))
        print("Max val is : "+str(max_val))
        pred[pred>0]=1
        pred[pred<=0]=0
```

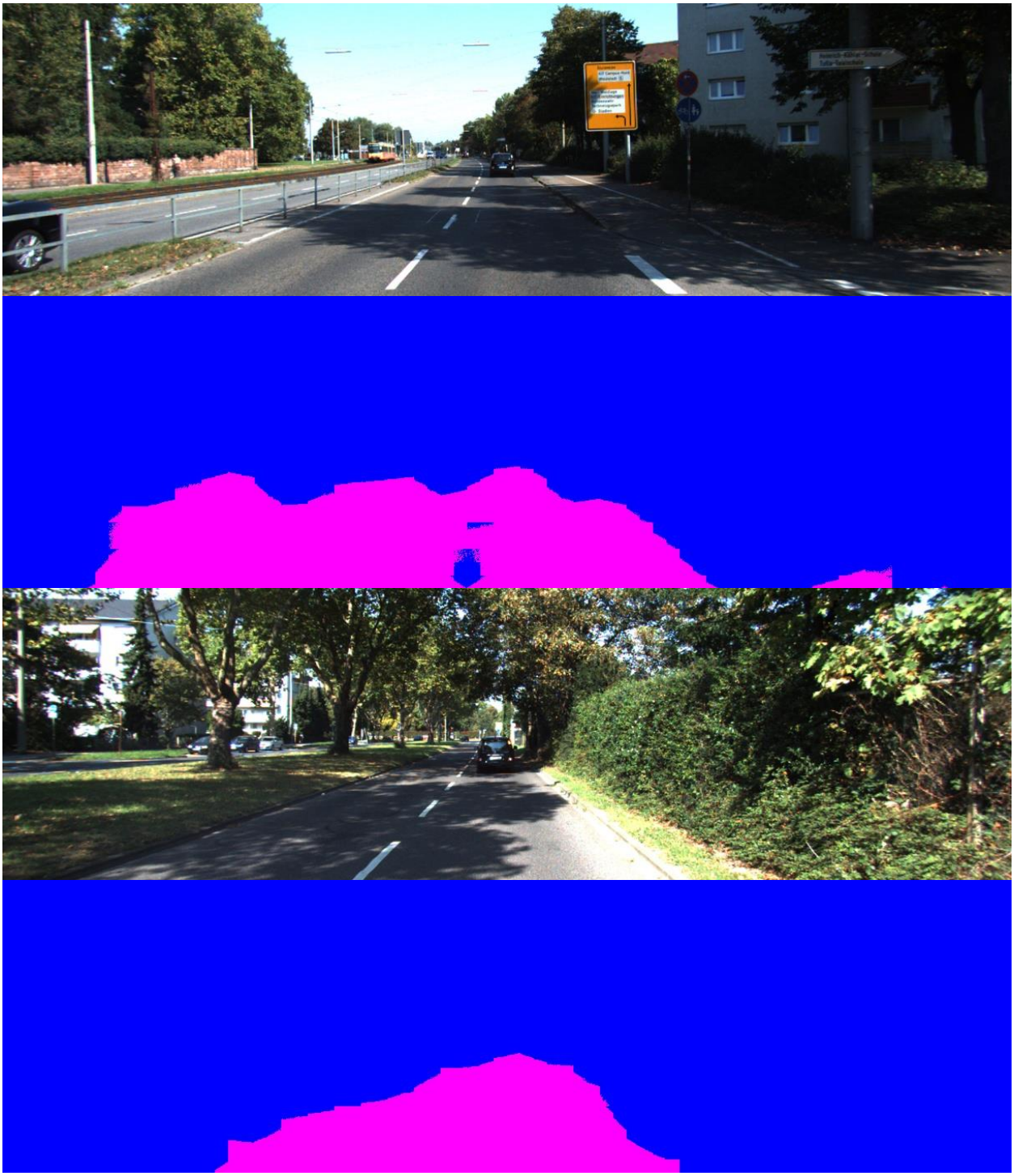
```
        print("Shape of pred is : "+str(pred.shape))
        pred_x = pred[0,:,:,0]
        print("Shape of pred_x is : "+str(pred_x.shape))
        predicted_image = np.empty(shape = [352,1216,3])
        for k in range(352):
            for j in range(1216):
                if pred_x[k,j] == 1:
                    predicted_image[k,j,:] = [255,0,255]
                else:
                    predicted_image[k,j,:] = [255,0,0]
        cv2.imwrite(f_np,predicted_image)
        TP = np.sum(np.logical_and(pred[:,:,:,0] == 1, batch_y == 1))
        TN = np.sum(np.logical_and(pred[:,:,:,0] == 0, batch_y == 0))
        FP = np.sum(np.logical_and(pred[:,:,:,0] == 1, batch_y == 0))
        FN = np.sum(np.logical_and(pred[:,:,:,0] == 0, batch_y == 1))
        tp+=TP
        tn+=TN
        fp+=FP
        fn+=FN
        i+=1
    no_of_batches = num_examples/BATCH_SIZE
    total_loss = total_loss/no_of_batches
    IOU = (float(tp))/(tp+fp+fn)
    print("TP is : "+str(tp))
    print("TN is : "+str(tn))
    print("FP is : "+str(fp))
    print("FN is : "+str(fn))
    return total_loss,IOU



with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    num_examples = len(train_data)
    train_loss = 0
    train_losses = []
    train_accuracies = []
    validation_losses = []
    validation_accuracies = []
    test_losses = []
    IOU_s = []
    validation_loss_min = 1
    print("[INFO] Training...")
    print()
    for i in range(EPOCHS):
        X_train, y_train = shuffle(train_data, train_labels)
```

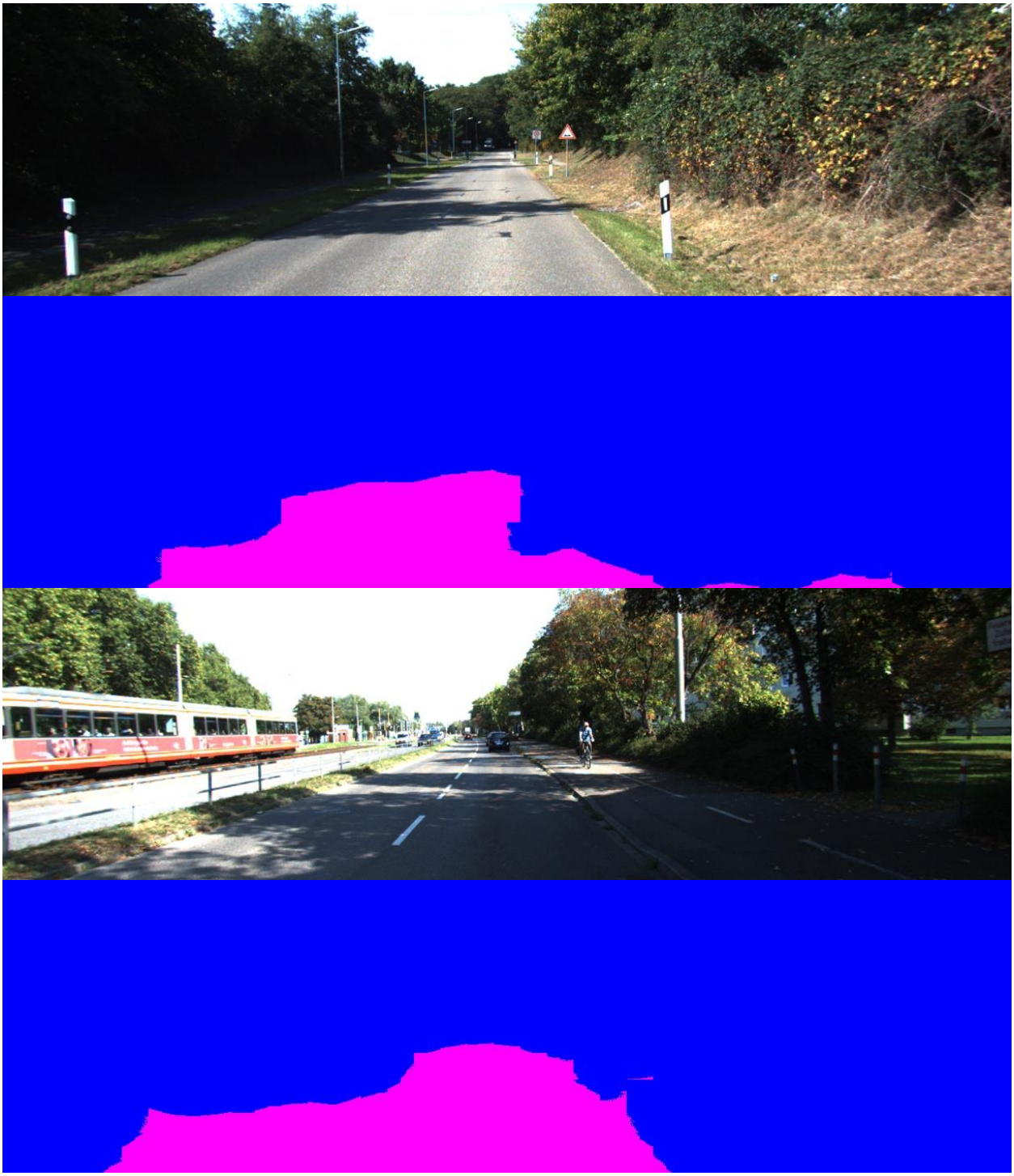
```
        for offset in range(0, num_examples, BATCH_SIZE):
            end = offset + BATCH_SIZE
            batch_x, batch_y = X_train[offset:end], y_train[offset:end]
            _,loss_val = sess.run([training_operation,loss_operation], feed_dict={x: batch_x, y: batch_y})
            train_loss+=loss_val
        no_of_batches = num_examples/BATCH_SIZE
        train_loss = train_loss/no_of_batches
        train_losses.append(train_loss)
        validation_loss_for_this_epoch,IOU = evaluate(validation_data, validation_labels)
        validation_losses.append(validation_loss_for_this_epoch)
        IOU_s.append(IOU)
        print()
        print("EPOCH {} ...".format(i+1))
        print("Training loss is :"+str(train_loss))
        print("Validation loss = {:.3f}".format(validation_loss_for_this_epoch))
        print("Validation IOU = {:.3f}".format(IOU))
        print()
        if validation_loss_for_this_epoch < validation_loss_min:
            name = './vggnet_fcn_{:.6f}'.format(validation_loss_for_this_epoch)
            saver.save(sess,name)
            validation_loss_min = validation_loss_for_this_epoch
    test_loss_for_this_epoch,IOU_test = evaluate1(test_data, test_labels)
    print("Test loss = {:.3f}".format(test_loss_for_this_epoch))
    print("Test IOU = {:.3f}".format(IOU_test))
    saver.save(sess, './lenet')
    print("Model saved")

epoch_nums = []
for i in range(1,EPOCHS+1):
    epoch_nums.append(i)

plt.style.use("ggplot")
plt.figure()
plt.plot(epoch_nums, train_losses,label="train_loss")
plt.plot(epoch_nums, validation_losses,label="val_loss")
plt.title("Training Loss, Validation Loss")
plt.xlabel("Epoch no.")
plt.ylabel("Loss")
plt.legend()
plt.savefig(args["plot"])

plt.style.use("ggplot")
plt.figure()
plt.plot(epoch_nums, IOU_s, label="IOU")
plt.xlabel("Epoch no.")
```

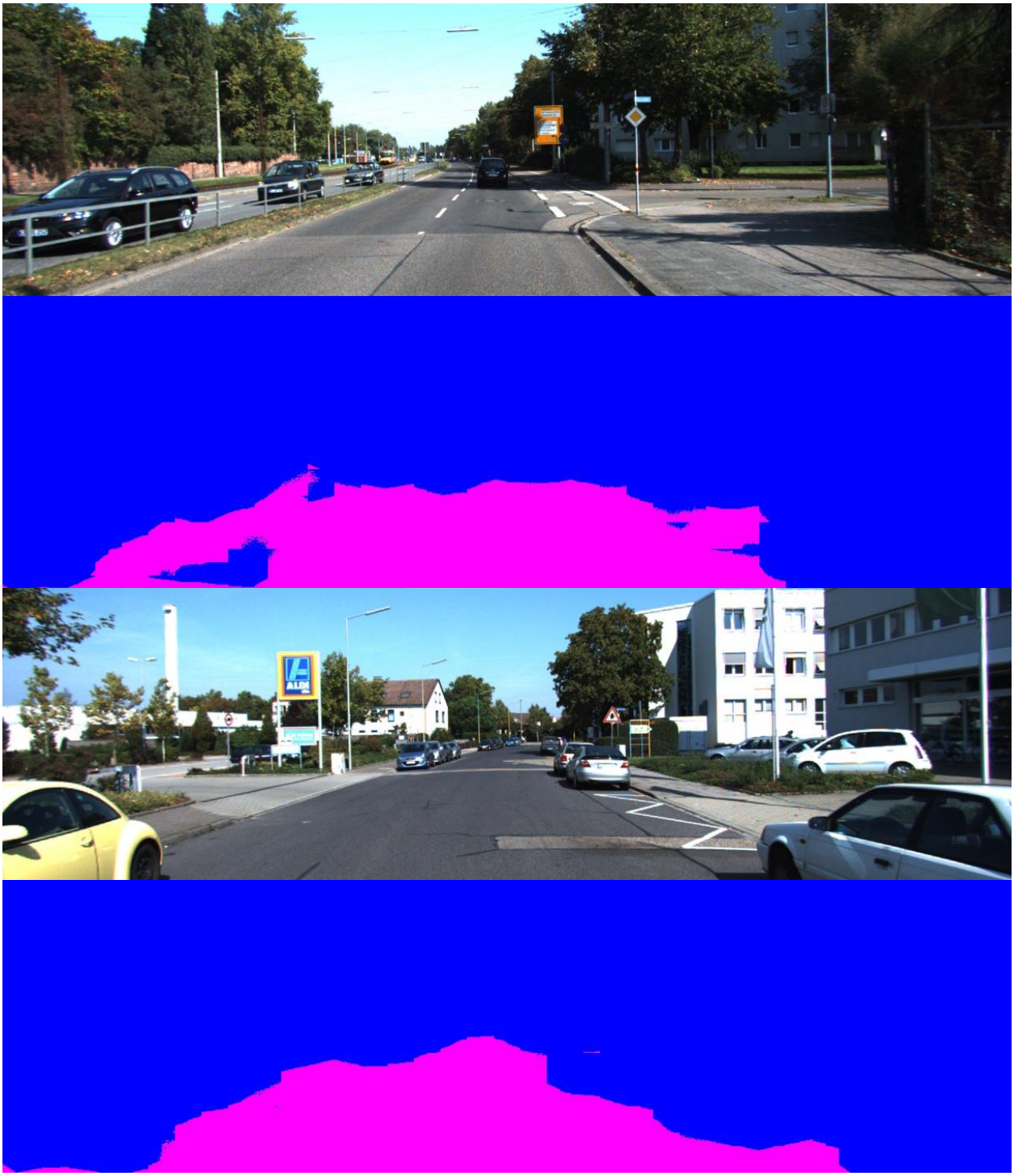
```
plt.ylabel("IOU")
plt.legend()
plt.savefig(args["IOU"])
```

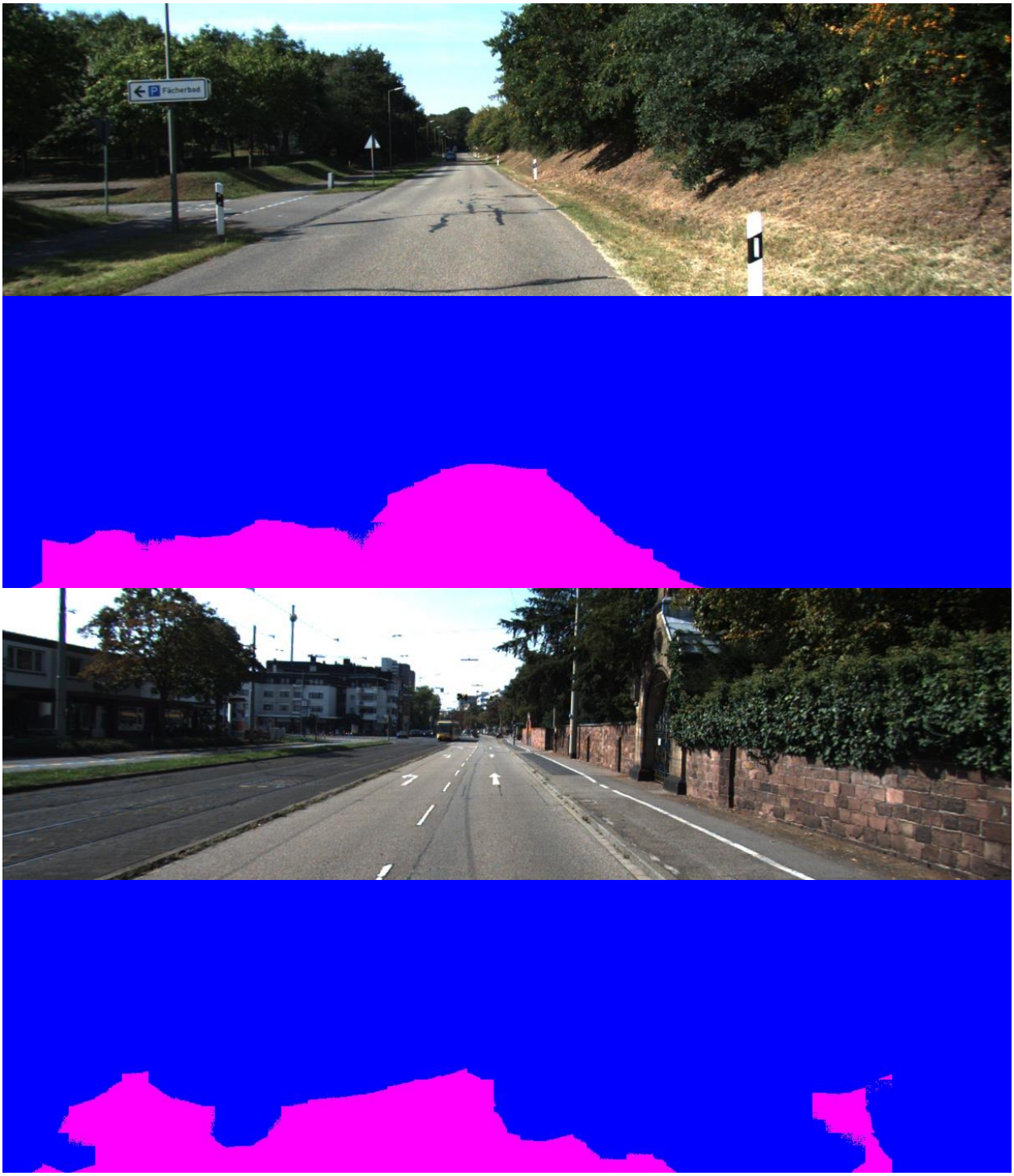
Appendix for Experiment 2:

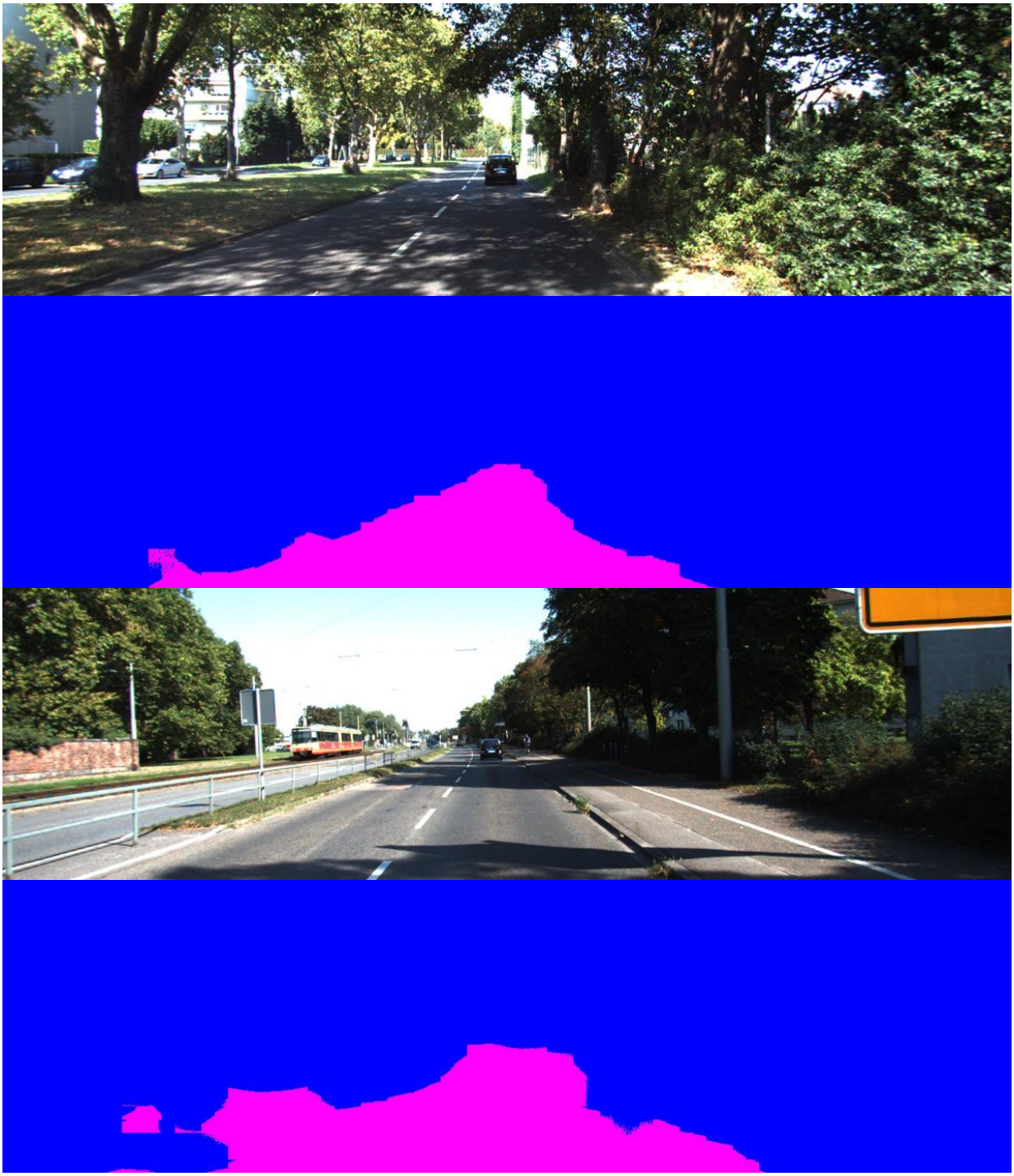


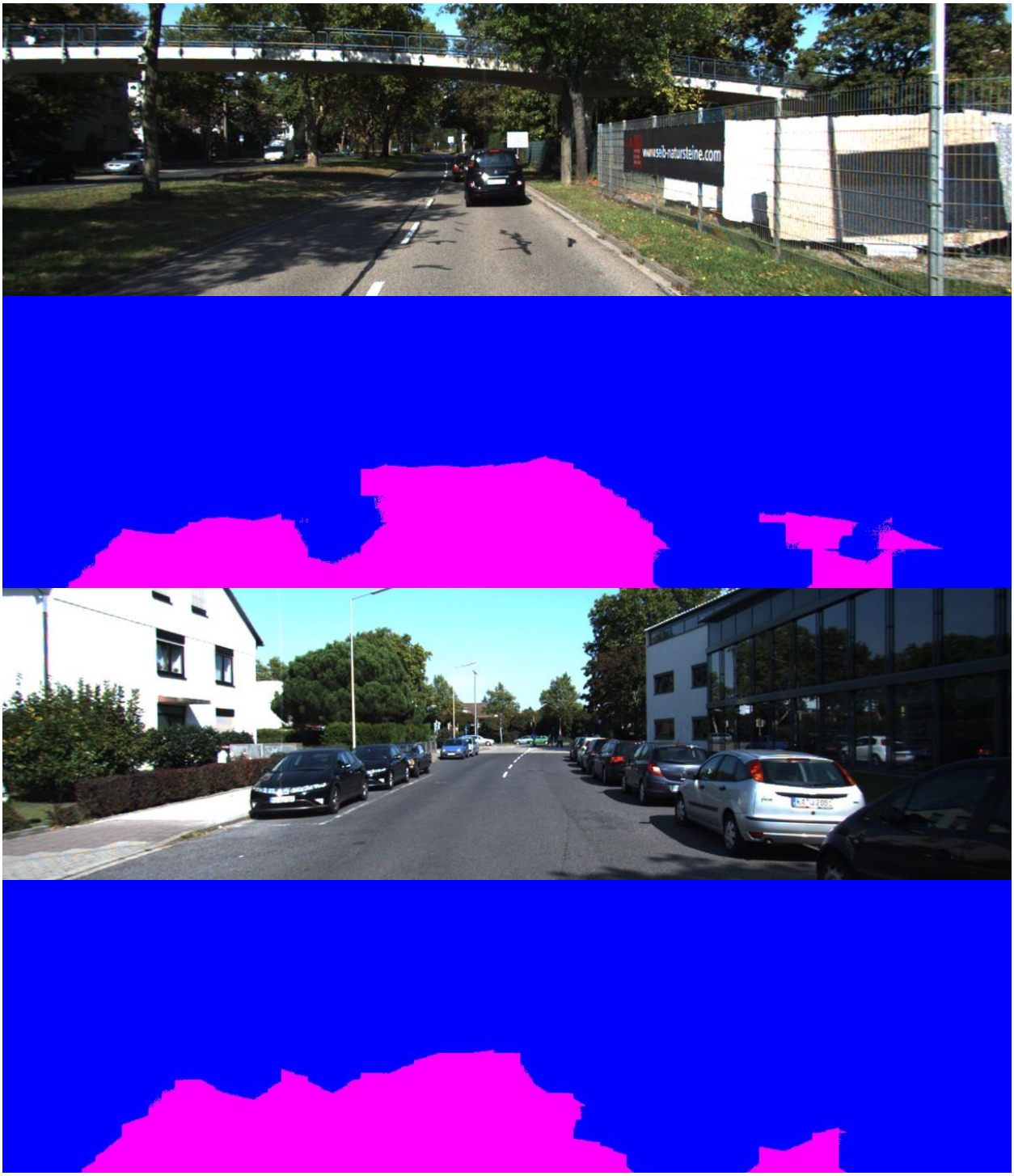


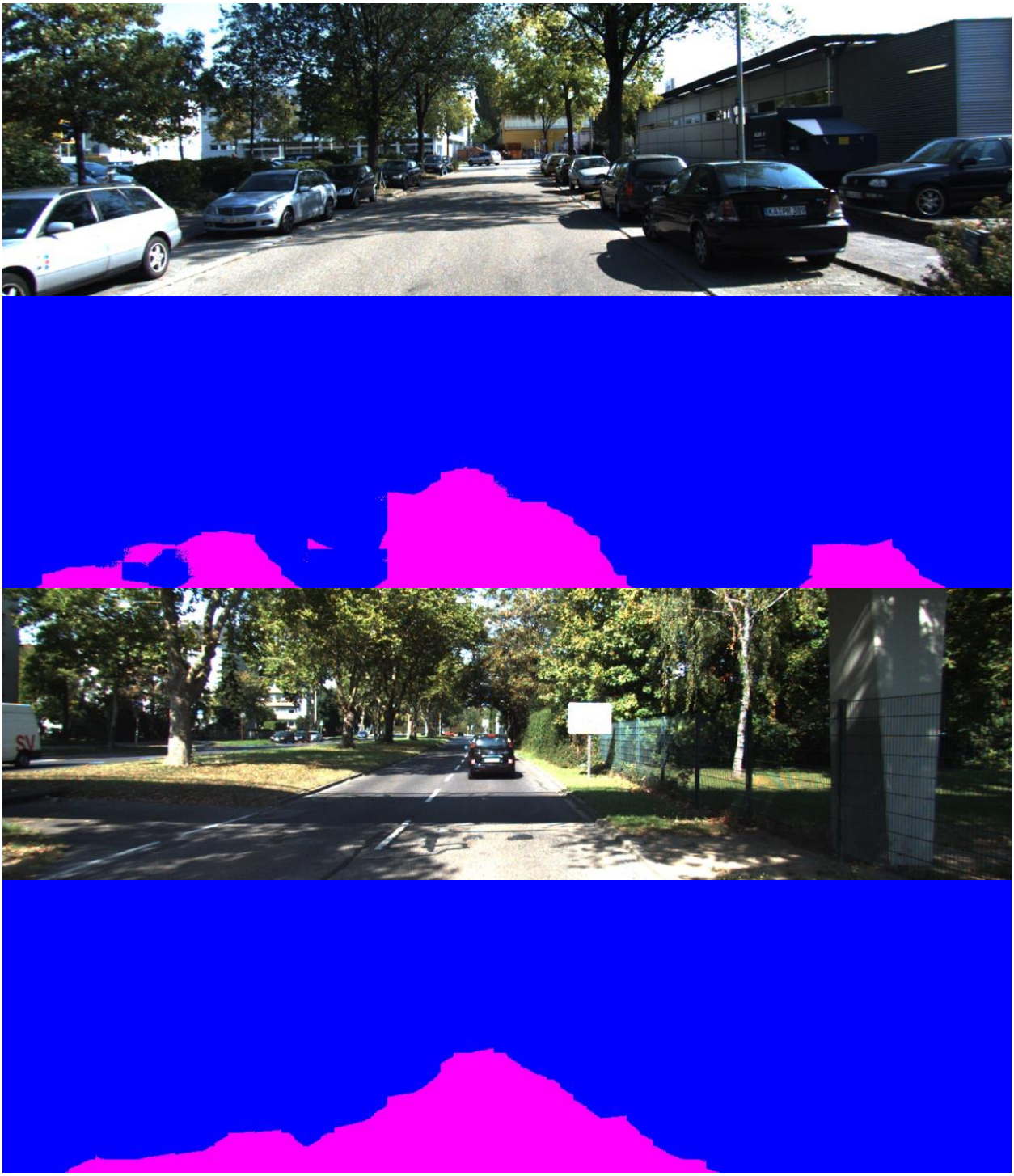


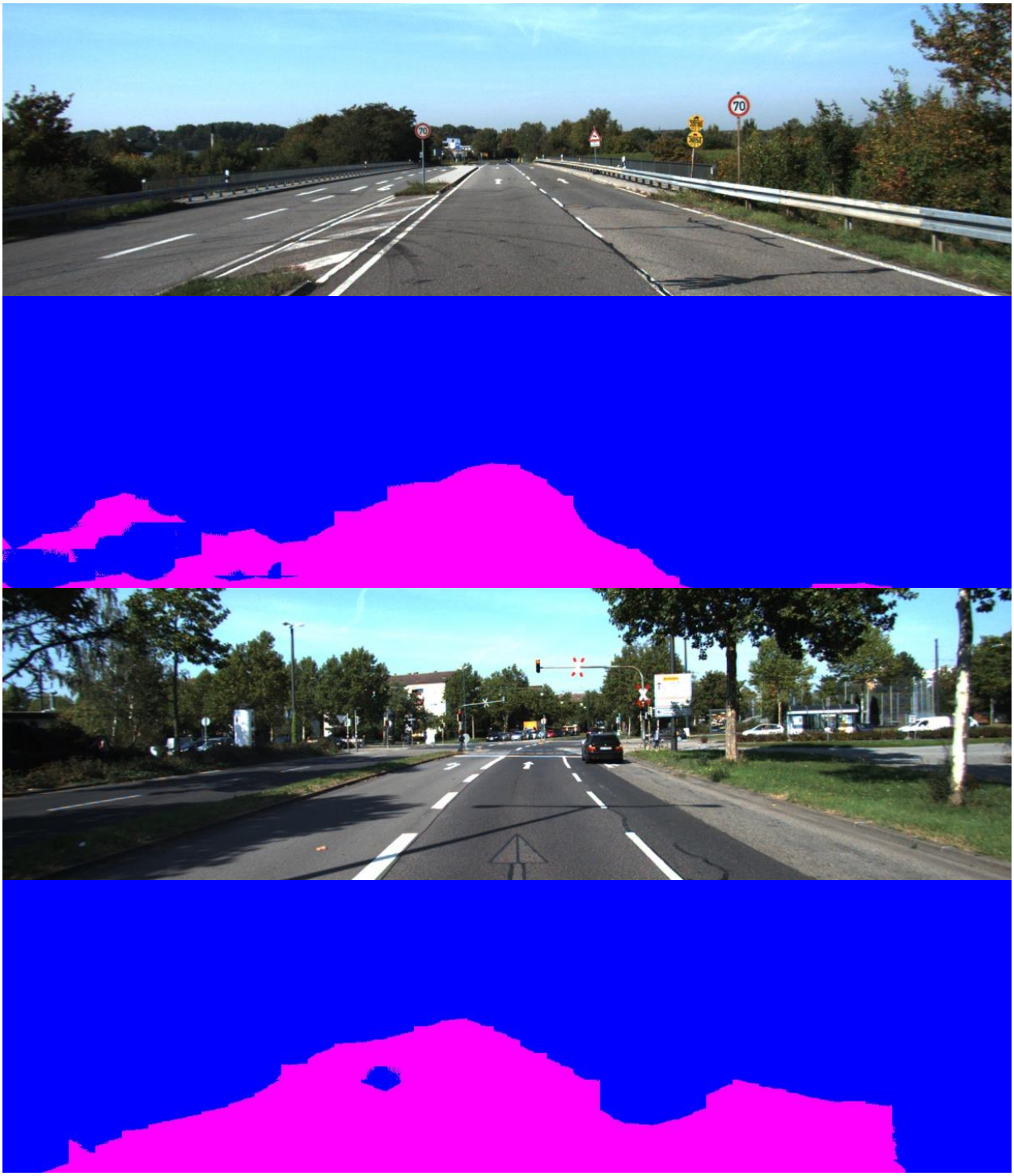

ALDI

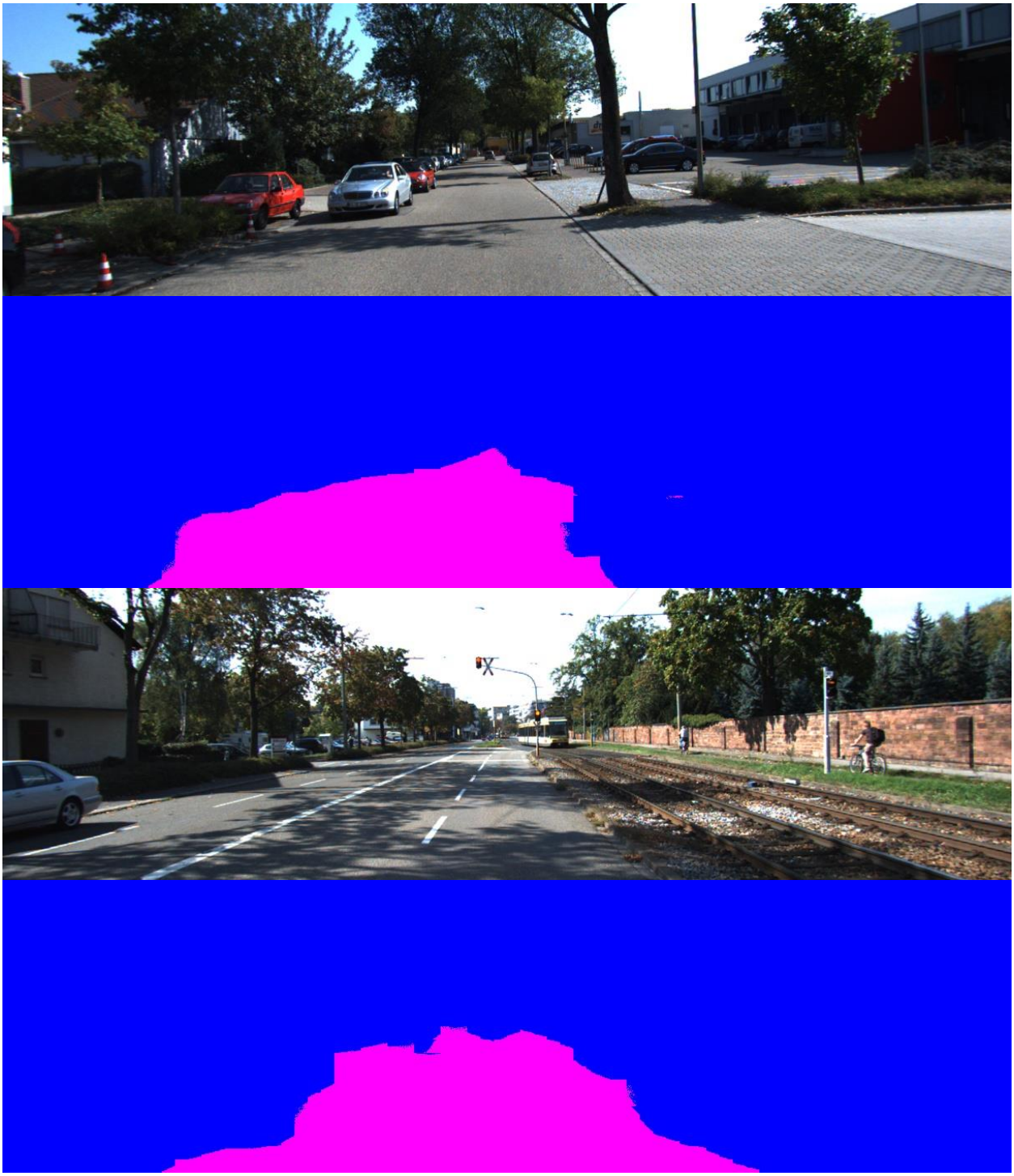

Fächerbad

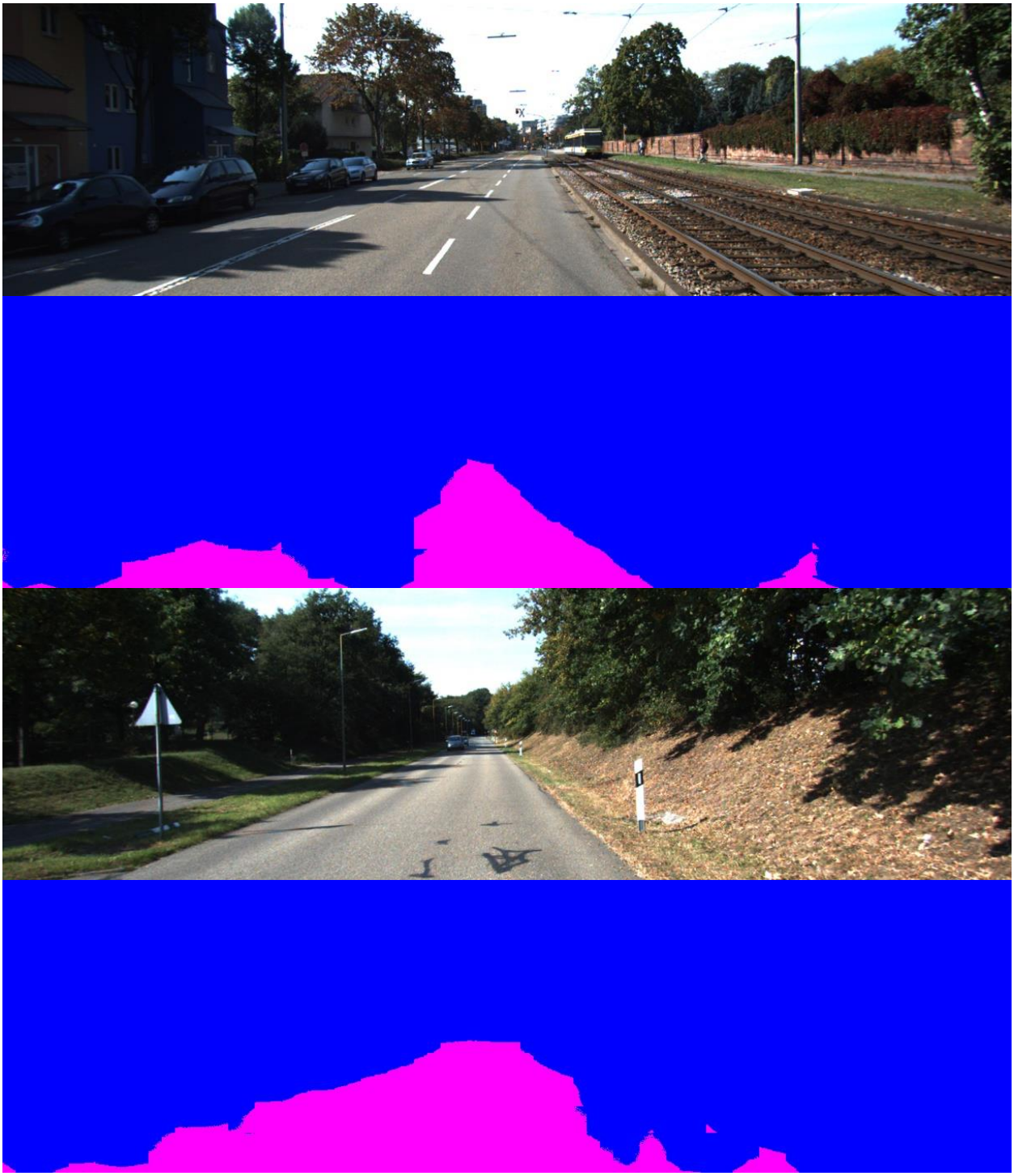


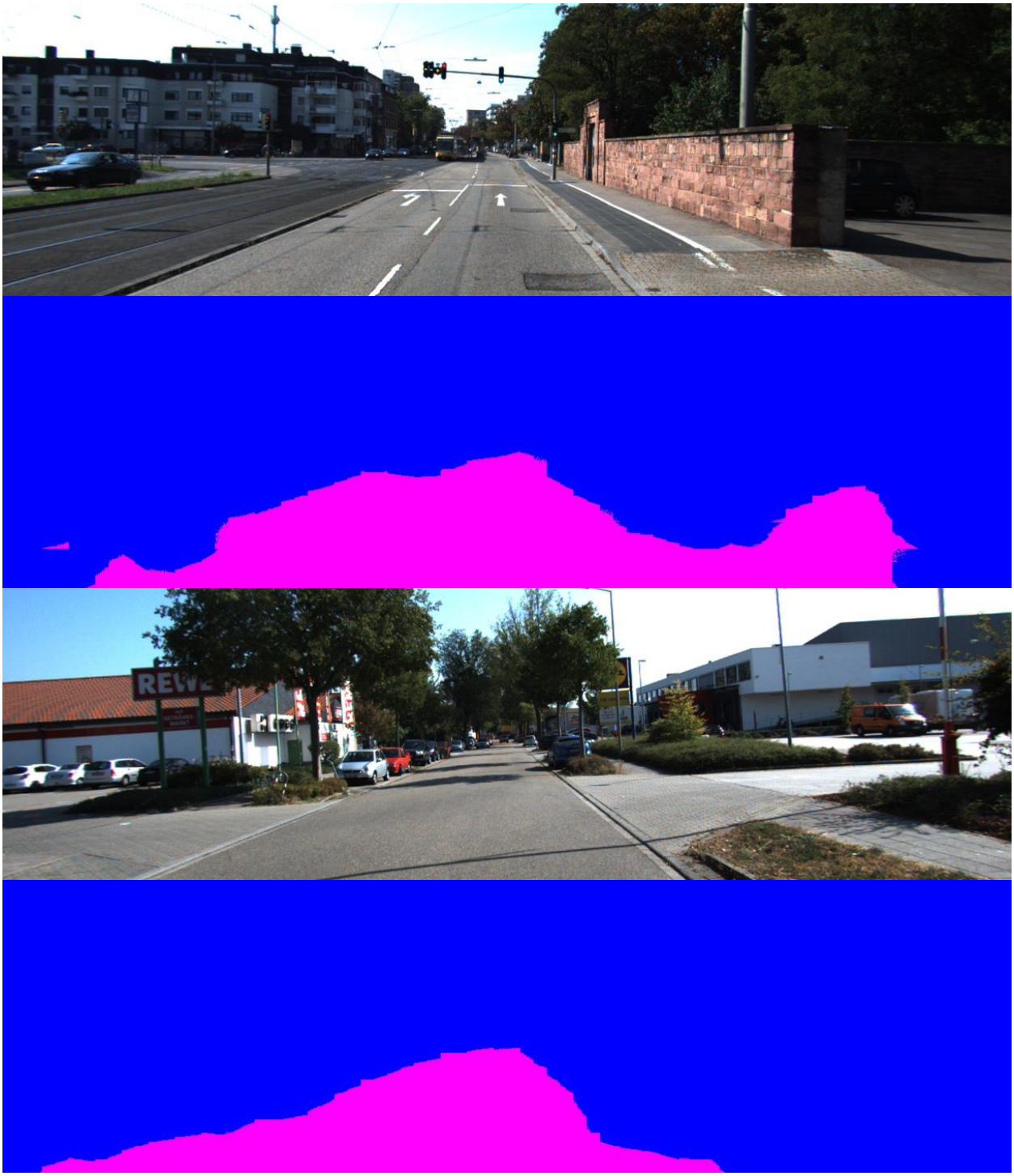

www.seb-natursteine.com

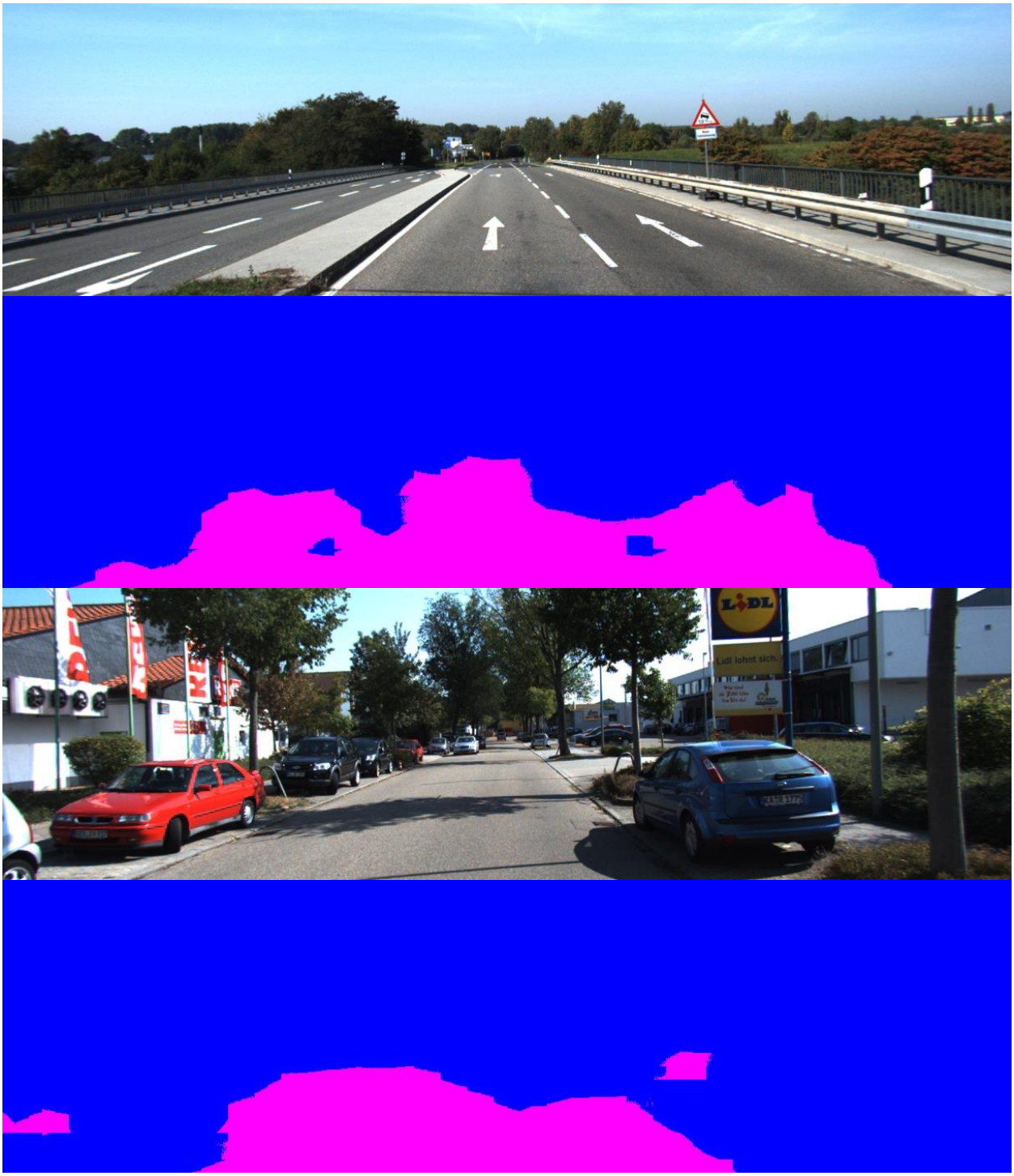


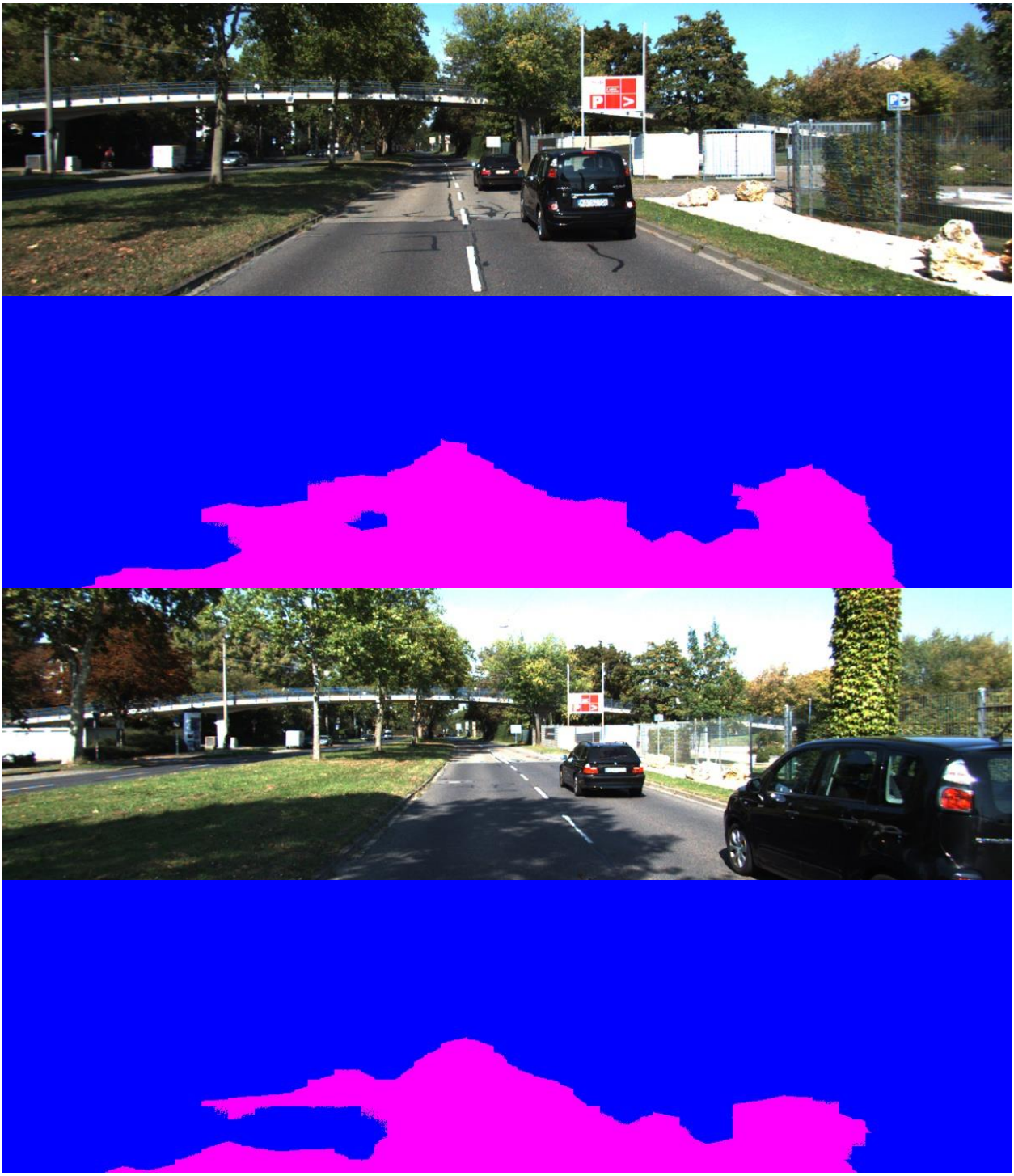

70
70

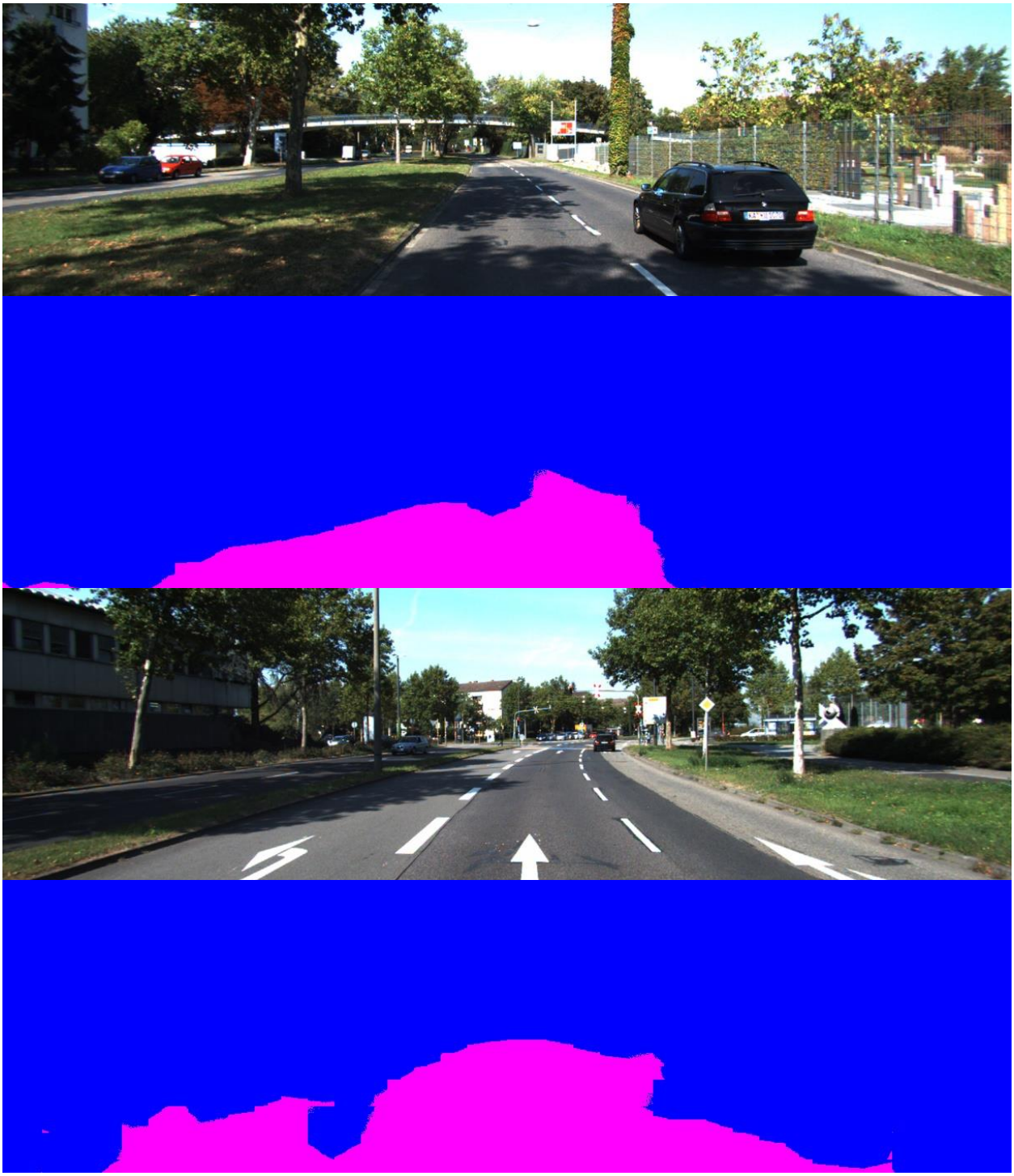


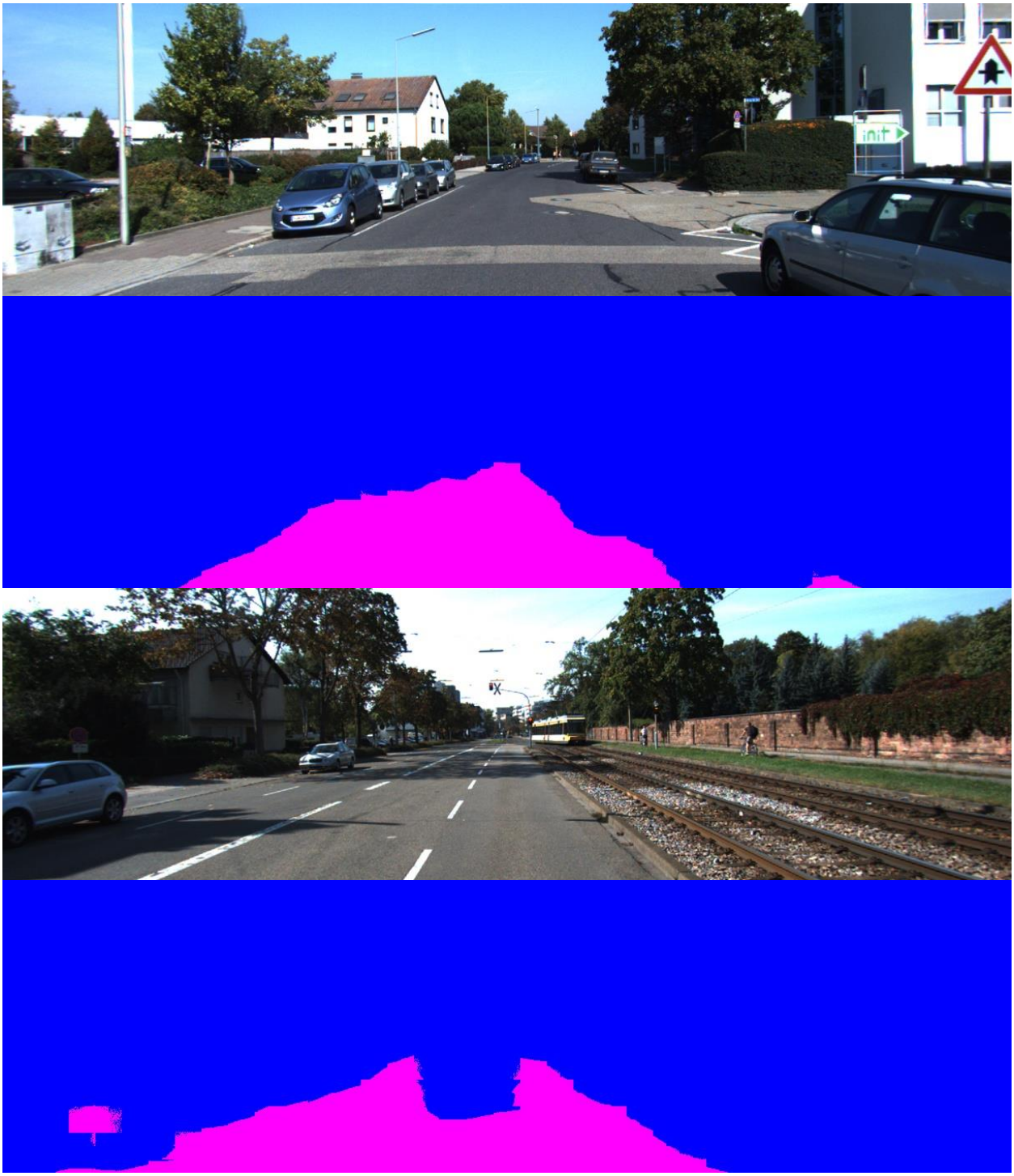


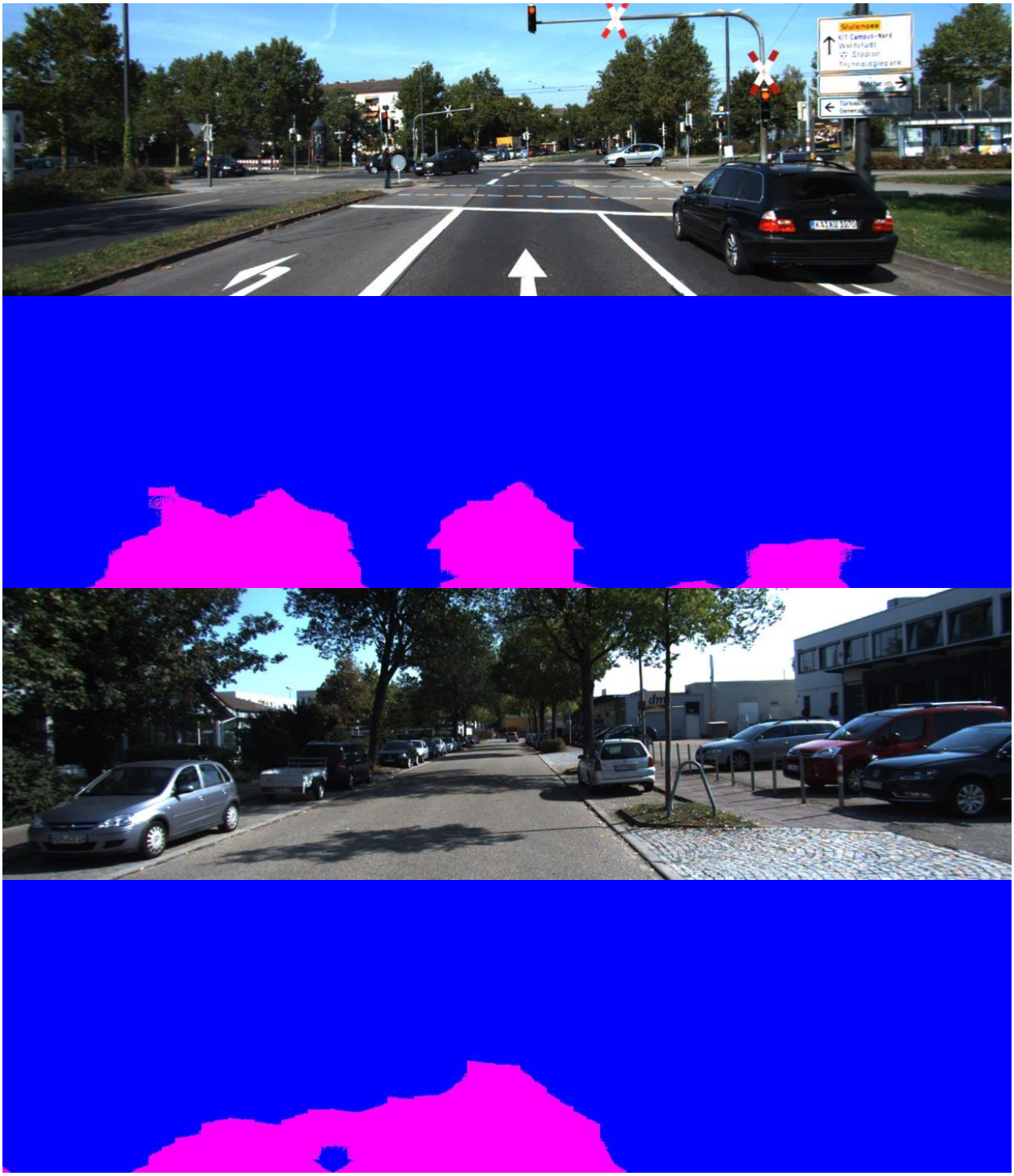

REWE

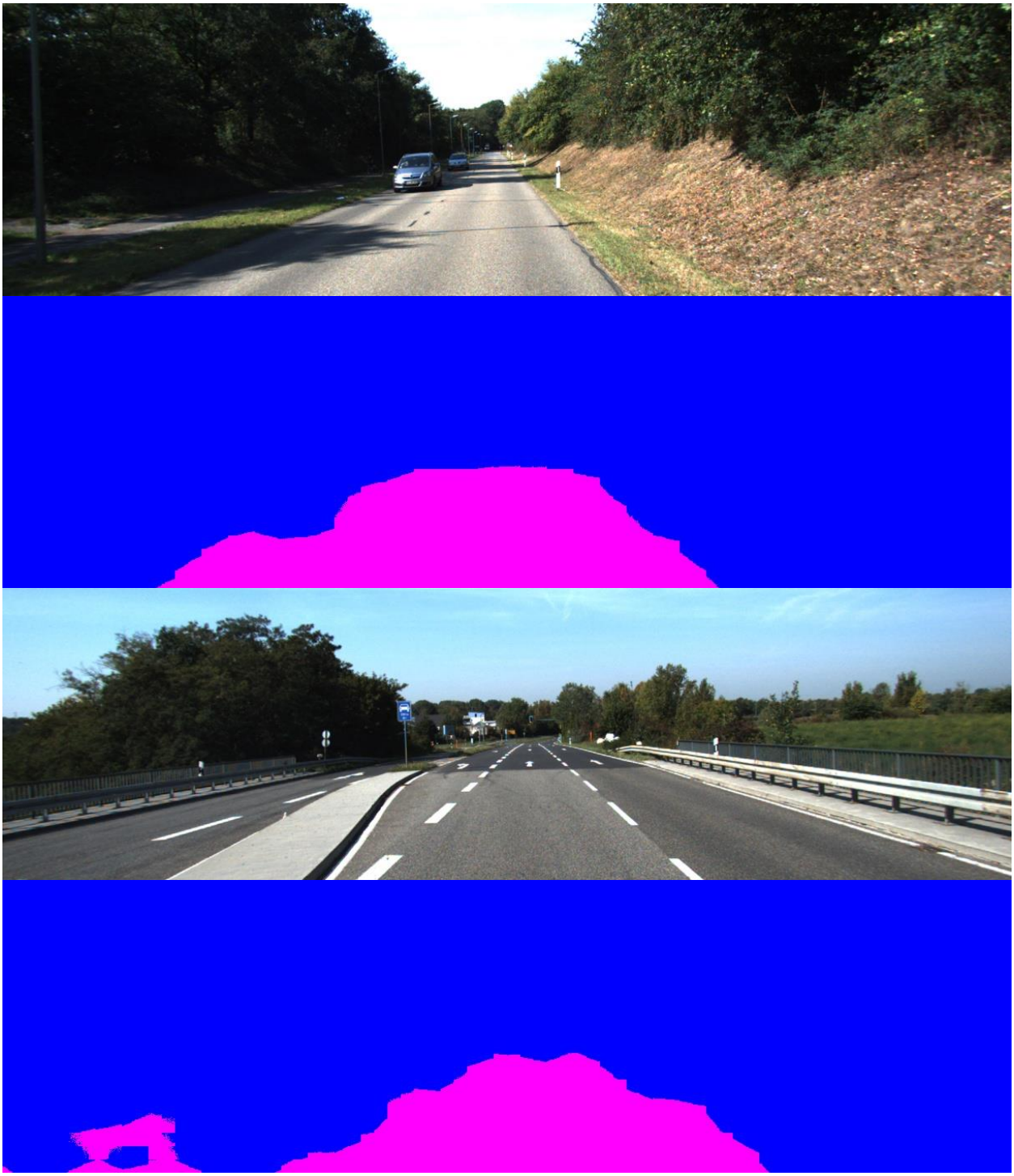

LIDL
Lidl lohnt sich

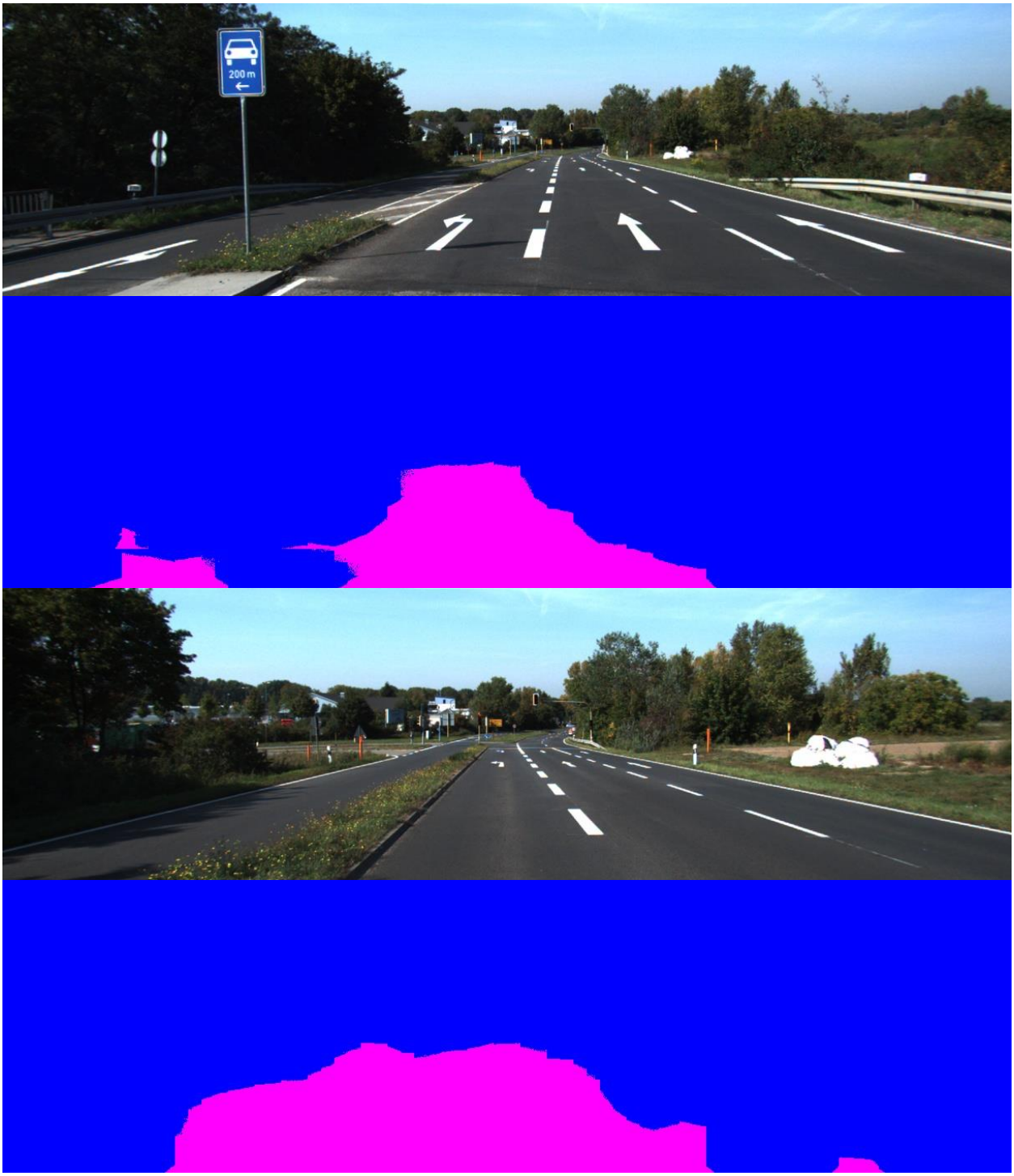


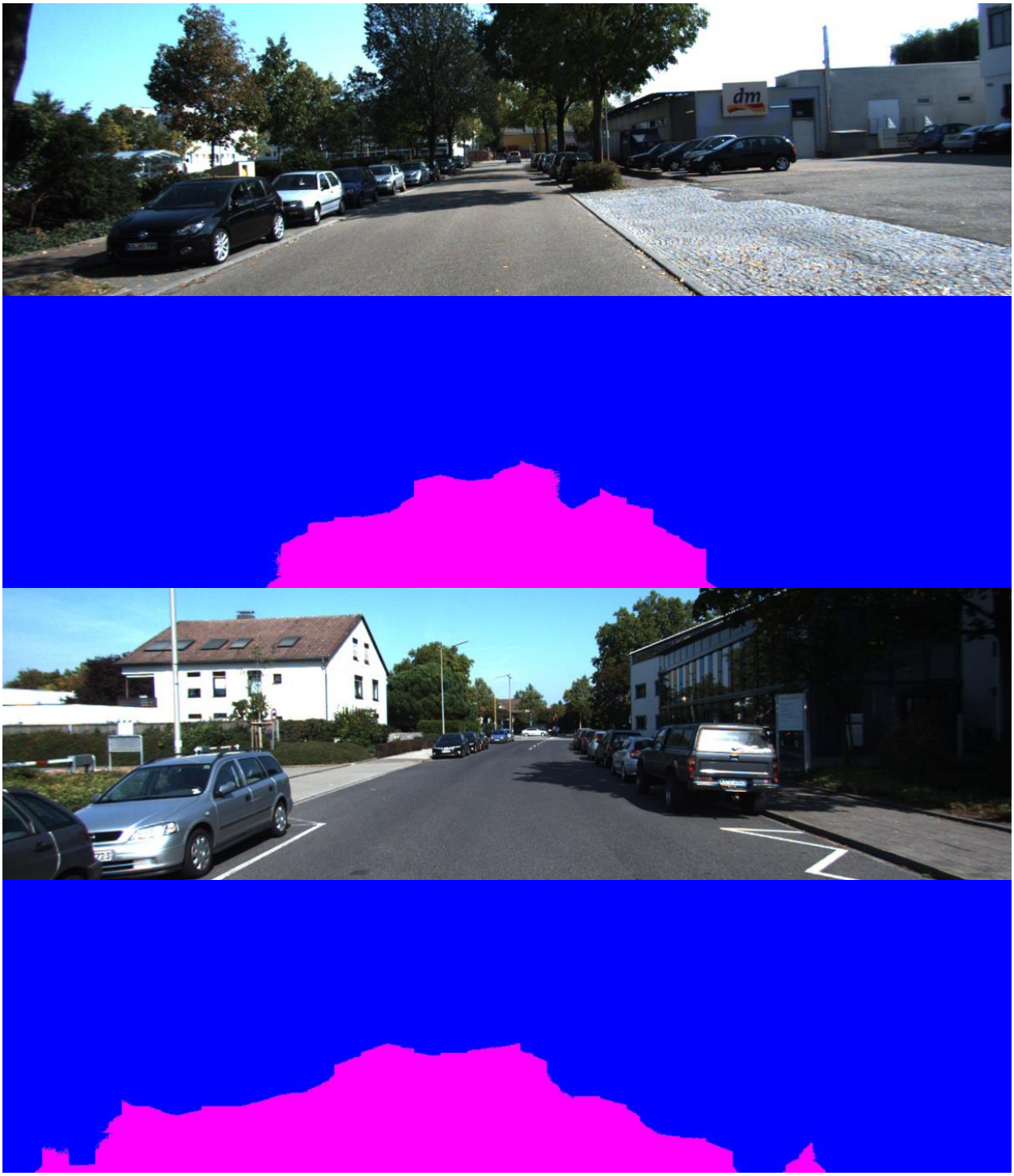


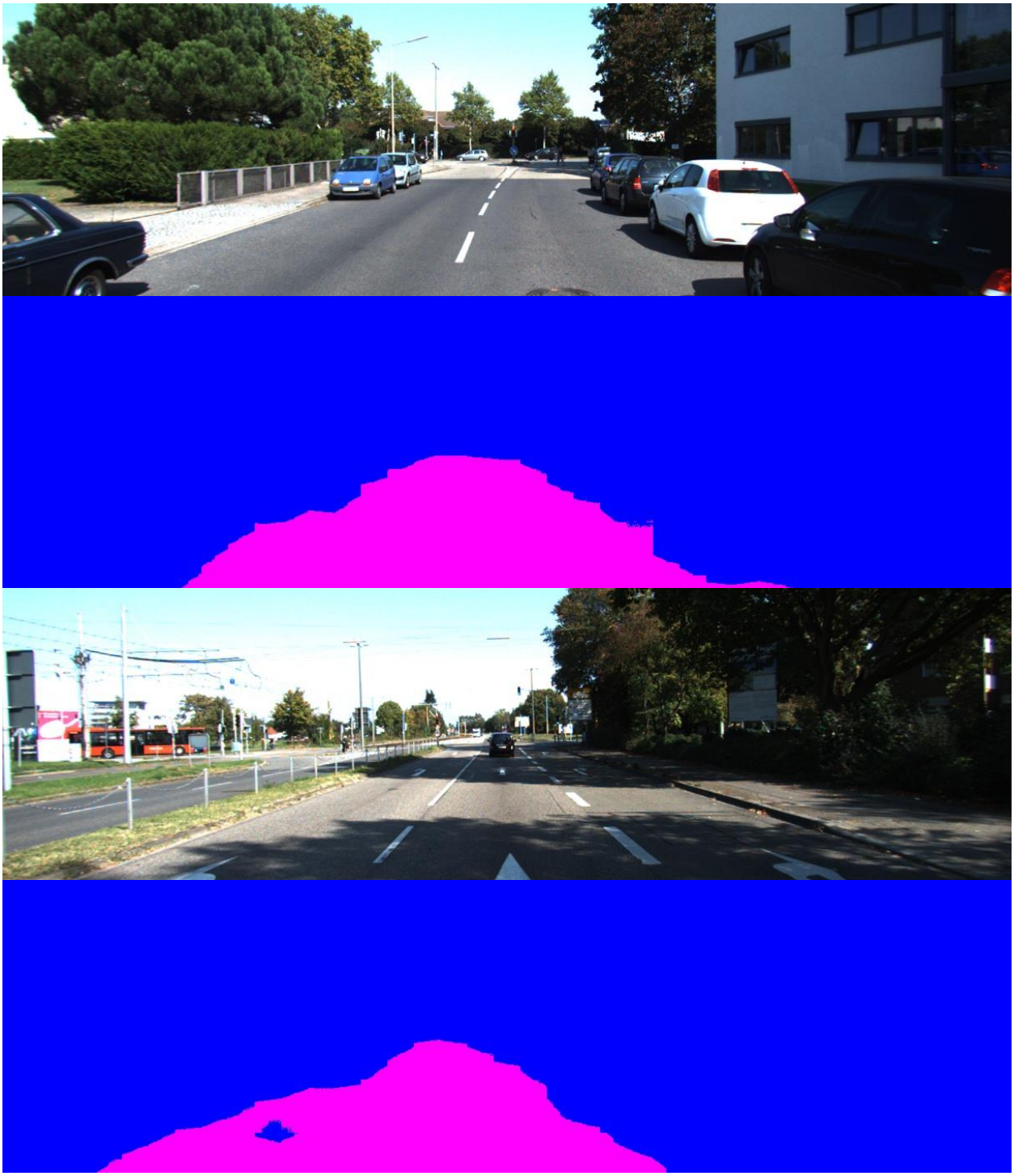

init

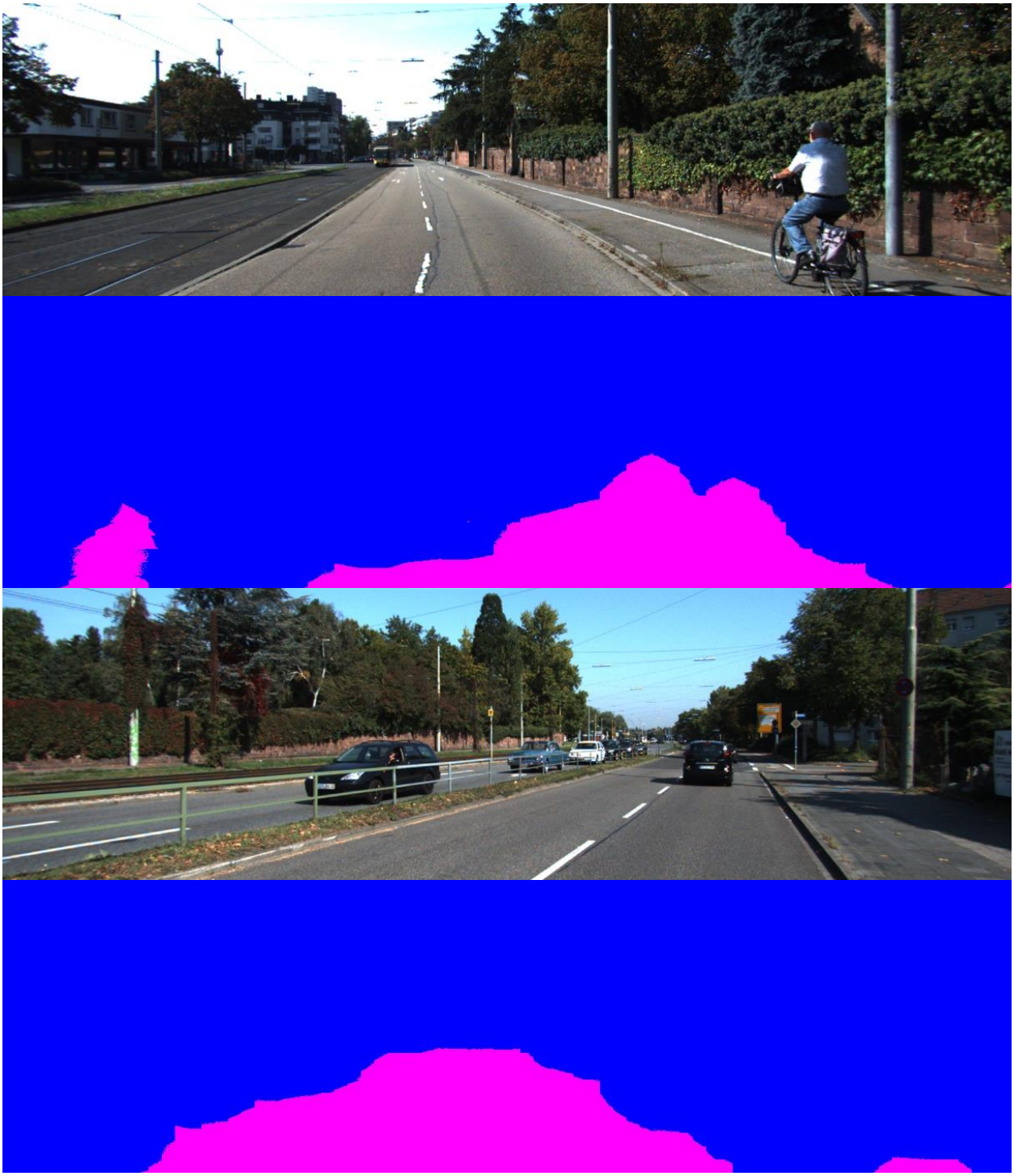

Studiensee
KIT Campus-Nord

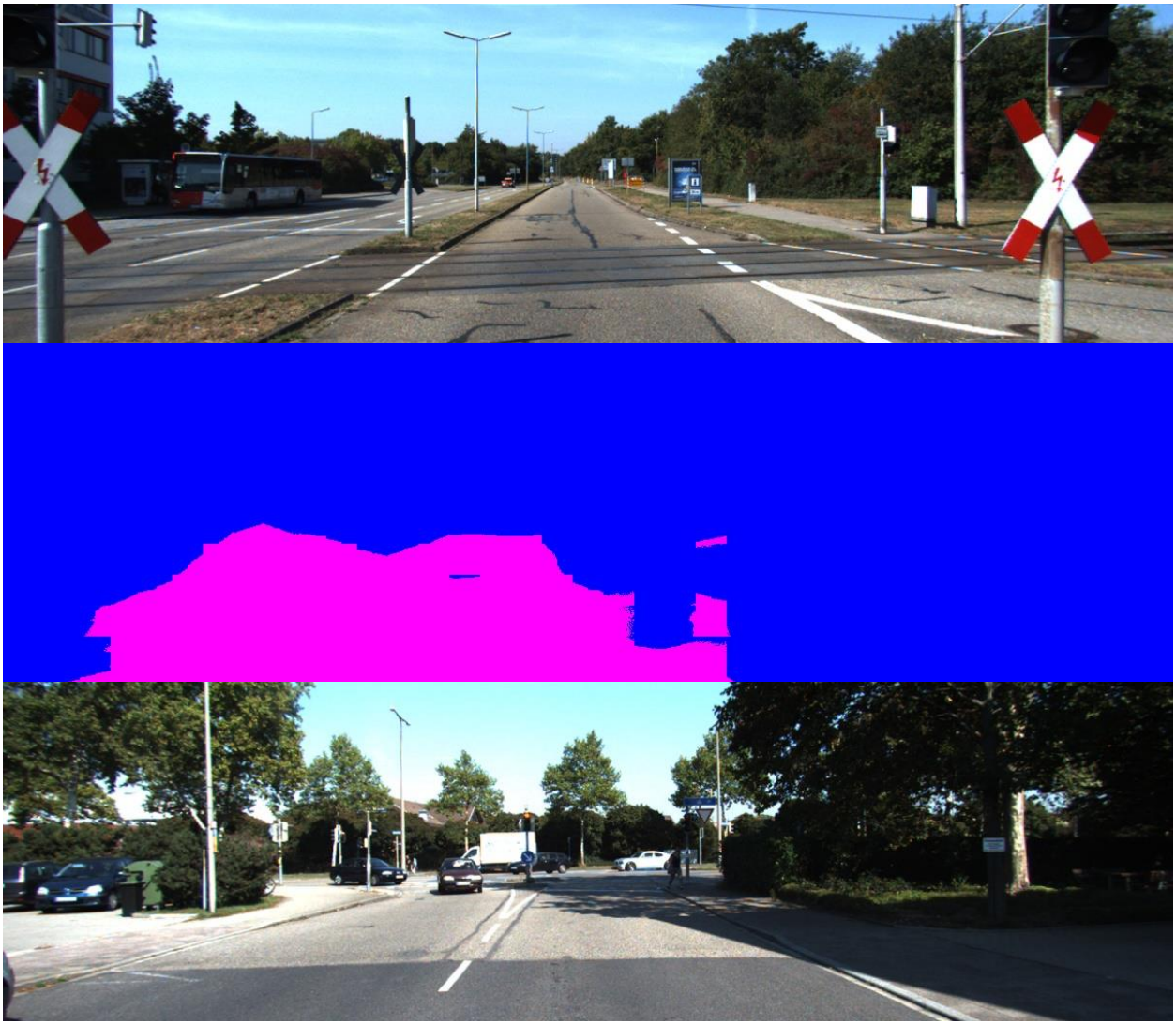



200 m

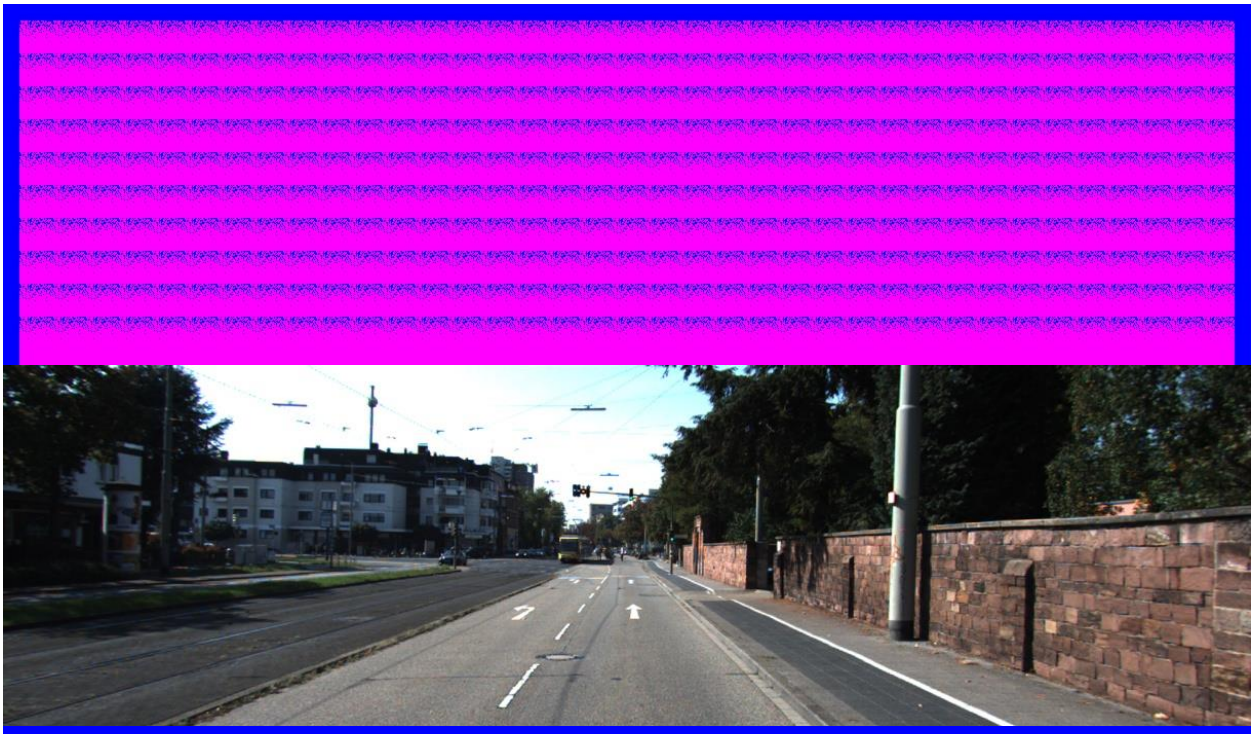

dm







Appendix for Experiment 3

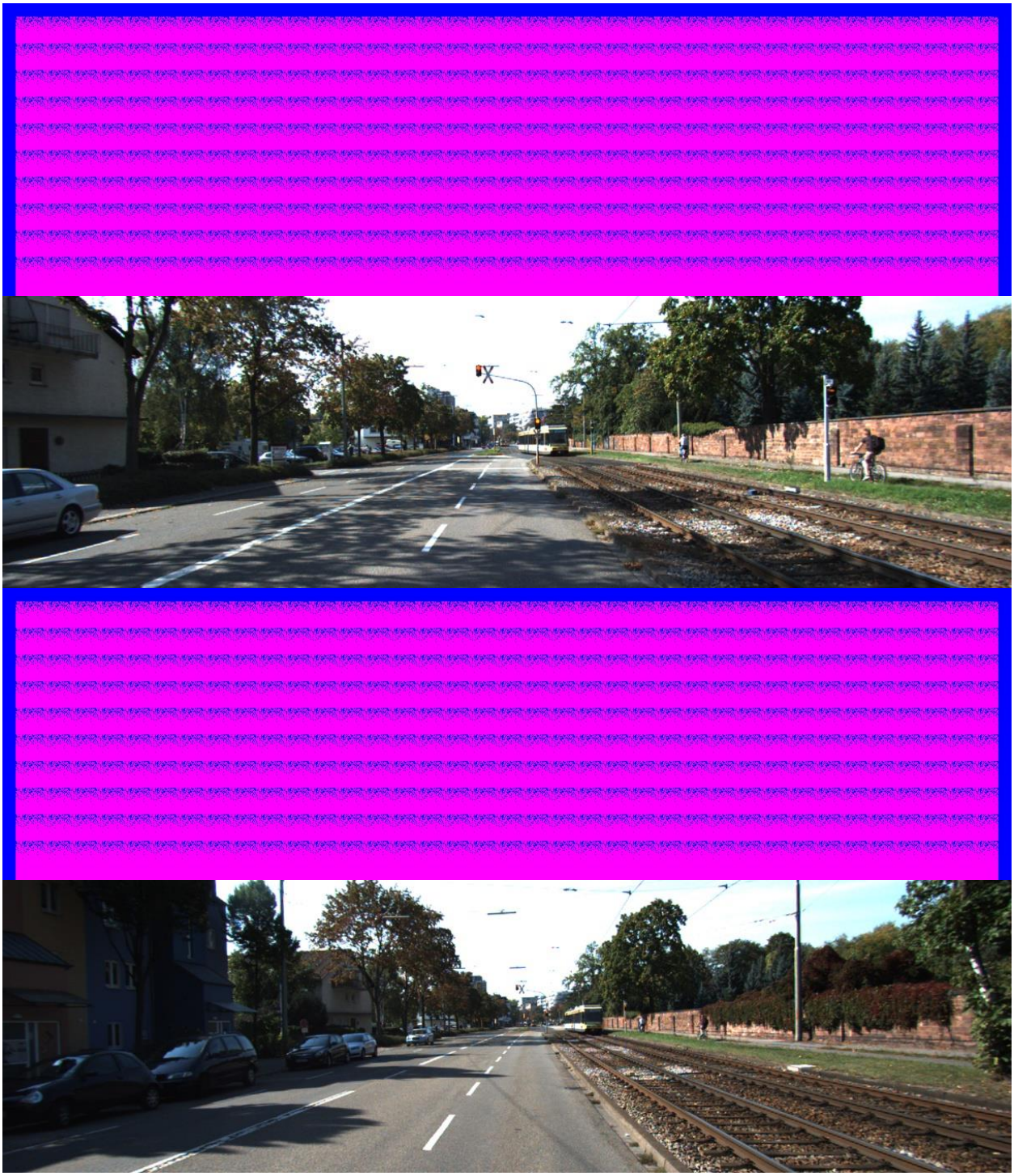


ALDI

Fächerbad

www.seib-natursteine.com

70
70



REWE

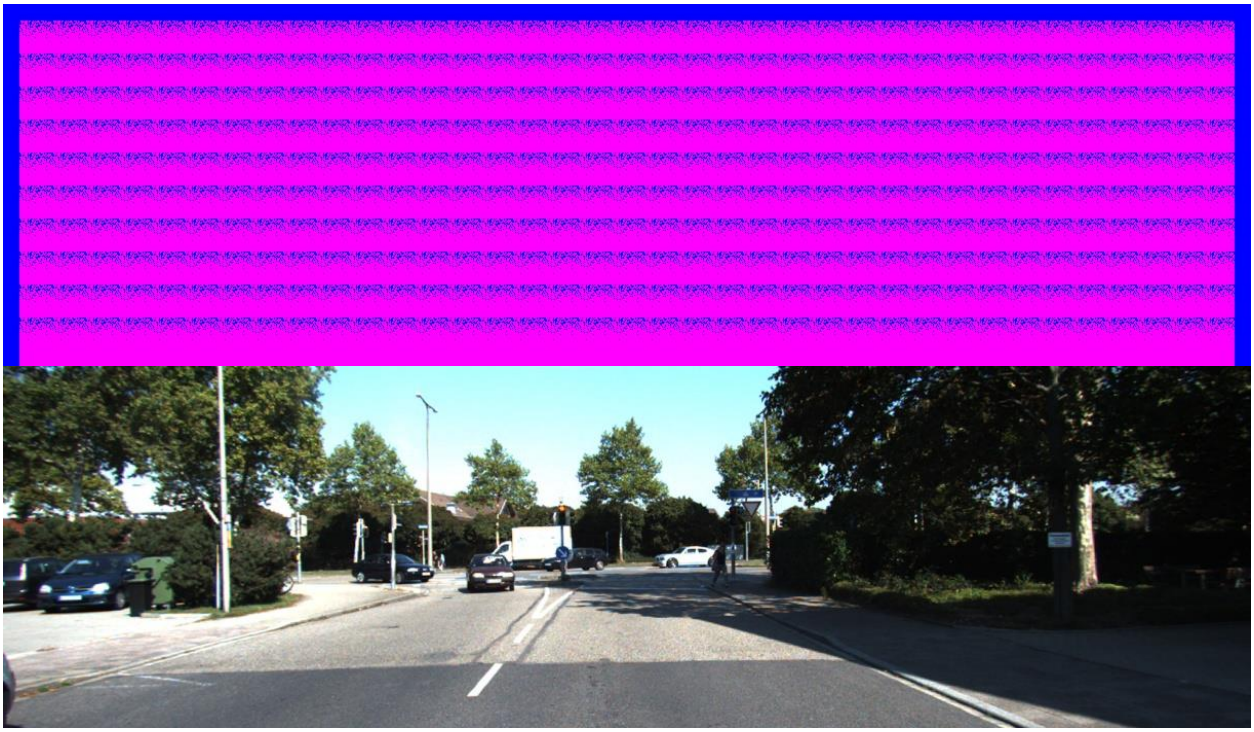
LIDL
Lidl lohnt sich

init

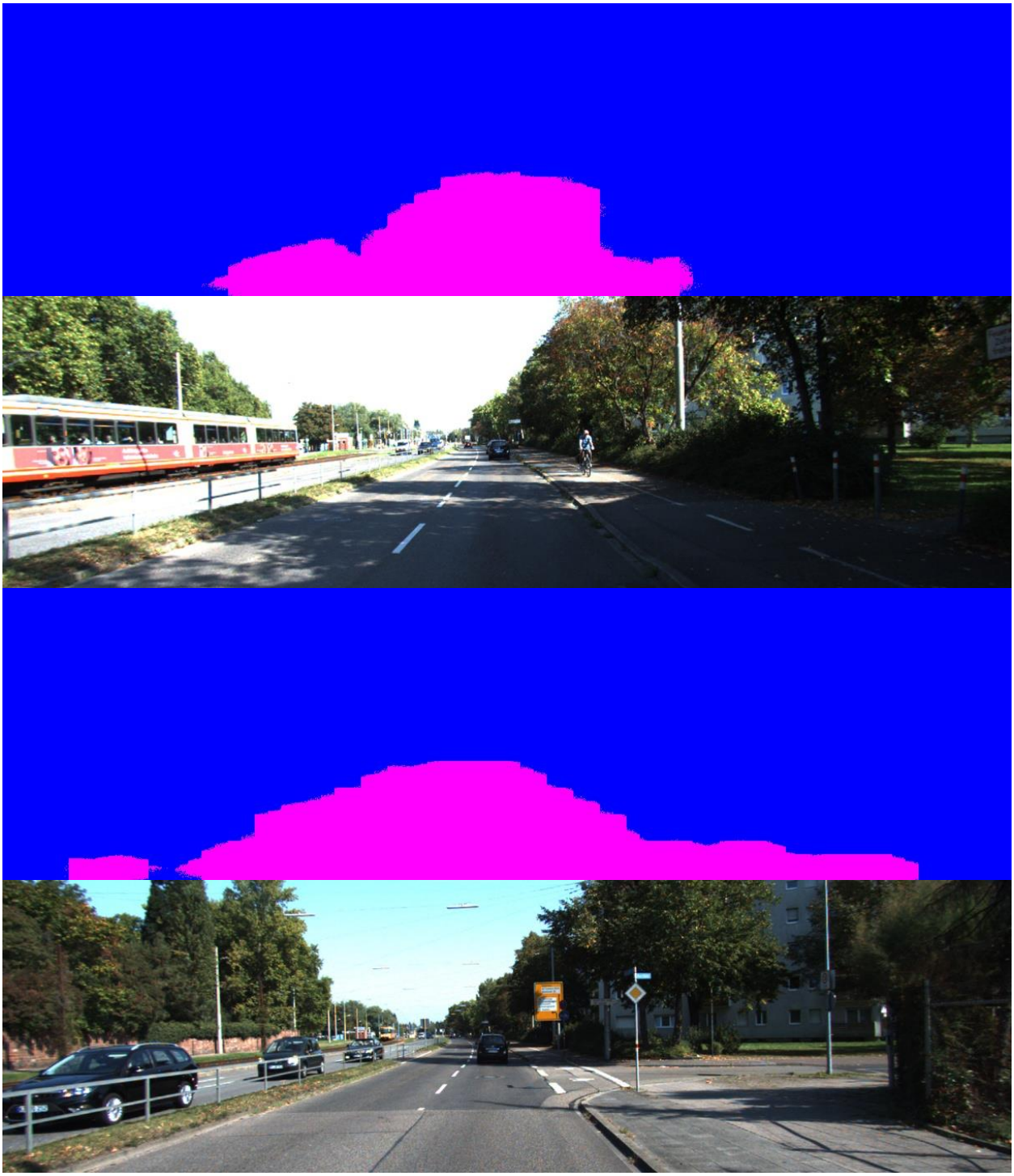
Stulensee

200 m

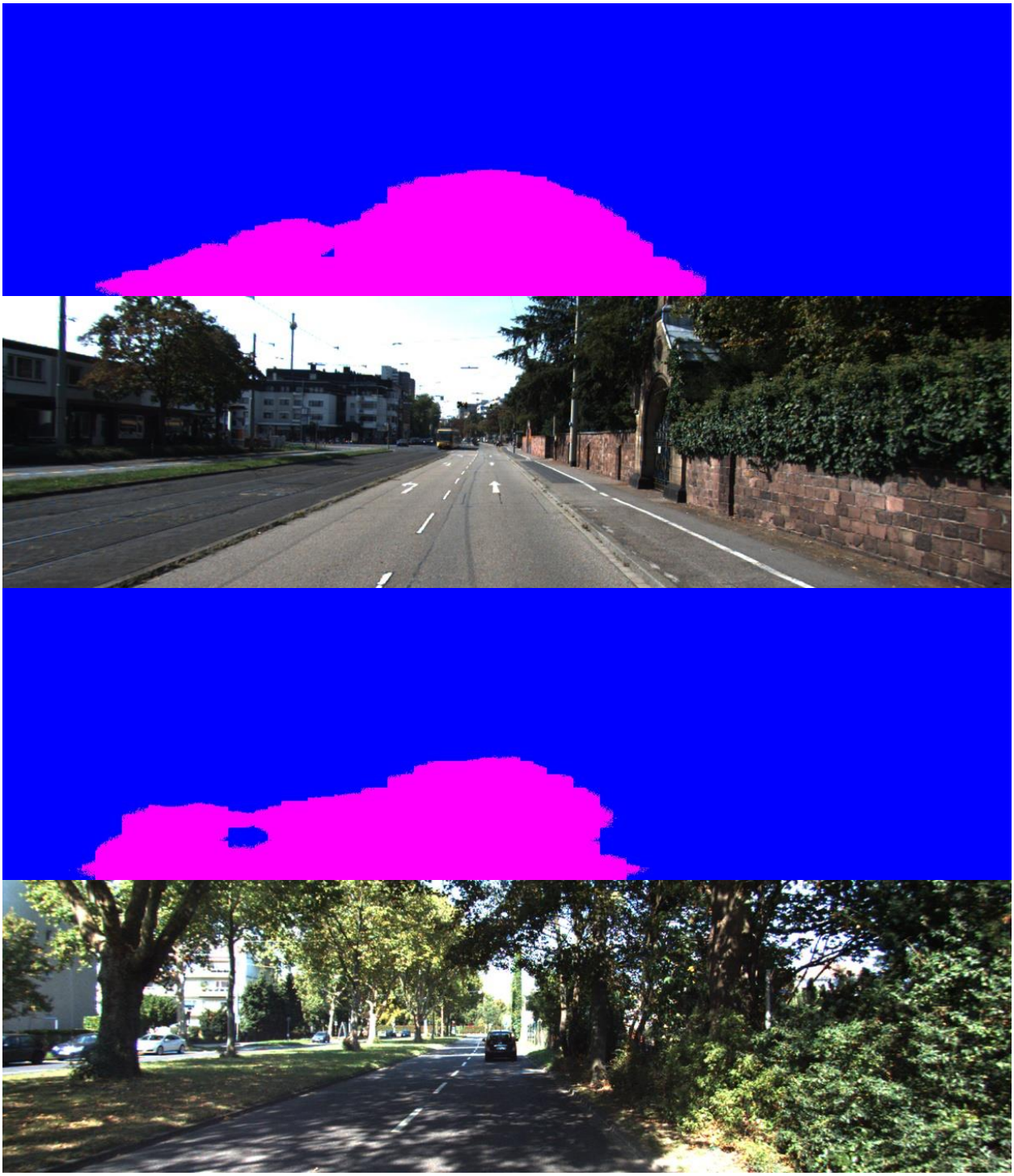
dm

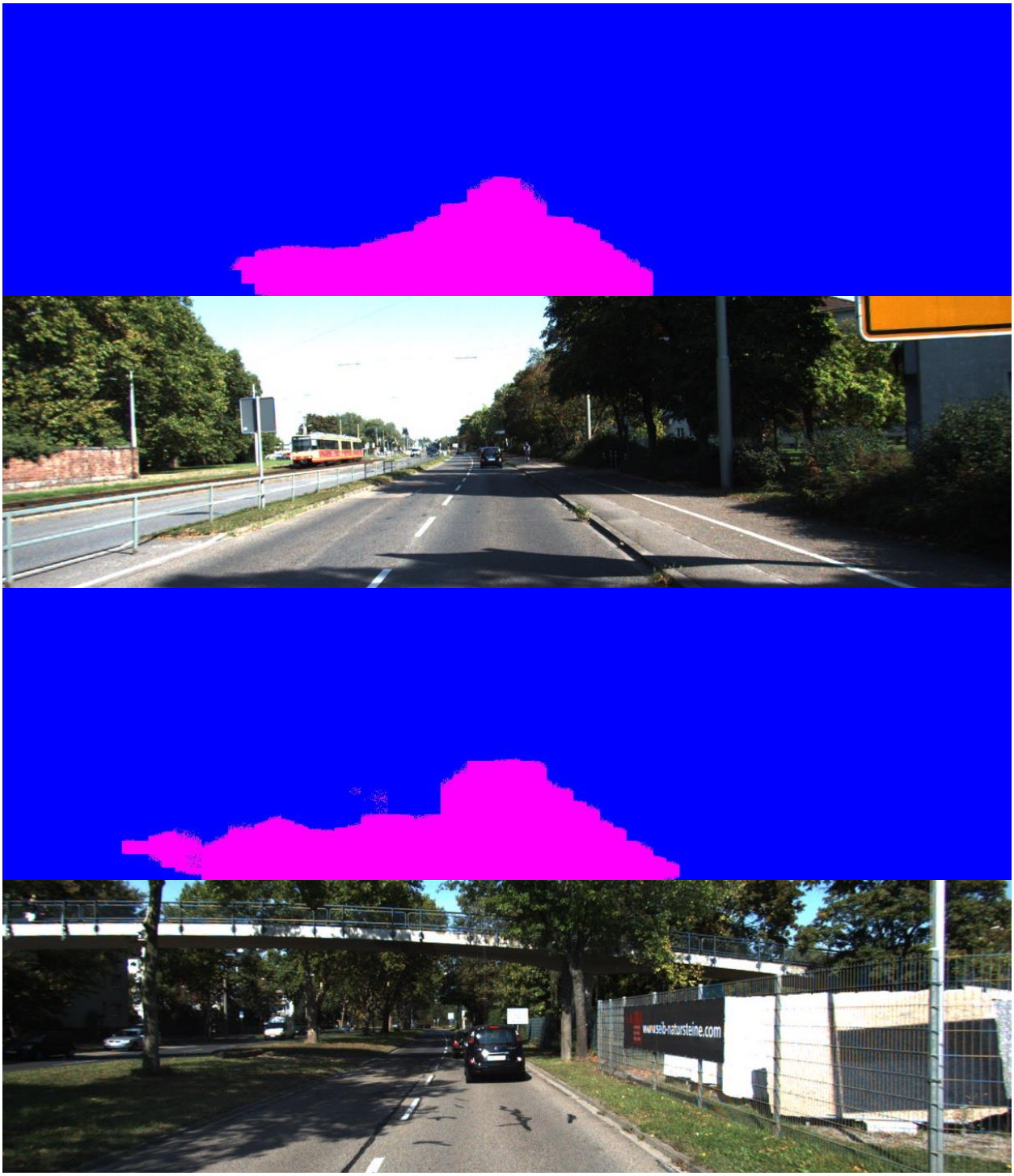


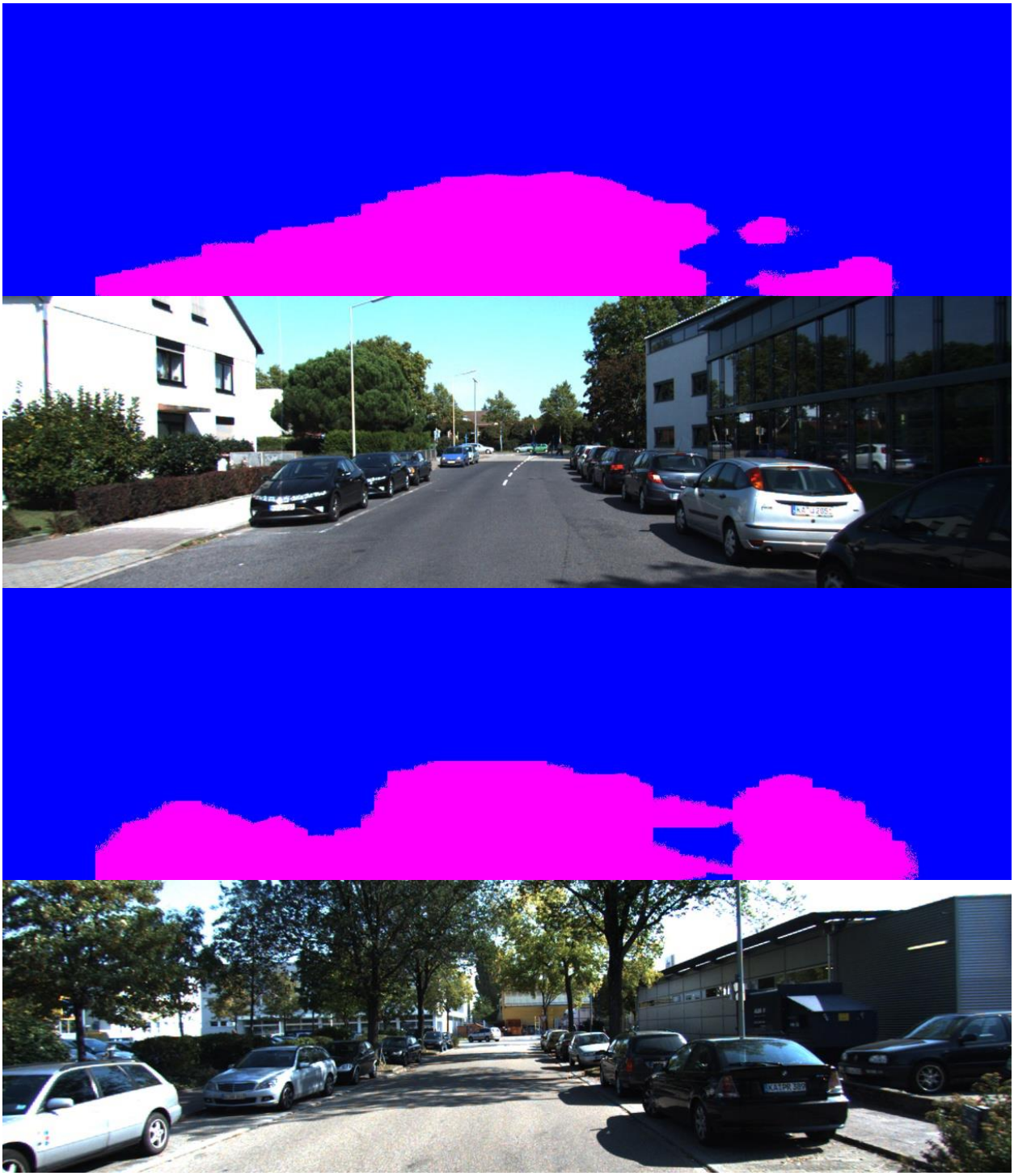
Appendix for Experiment 4

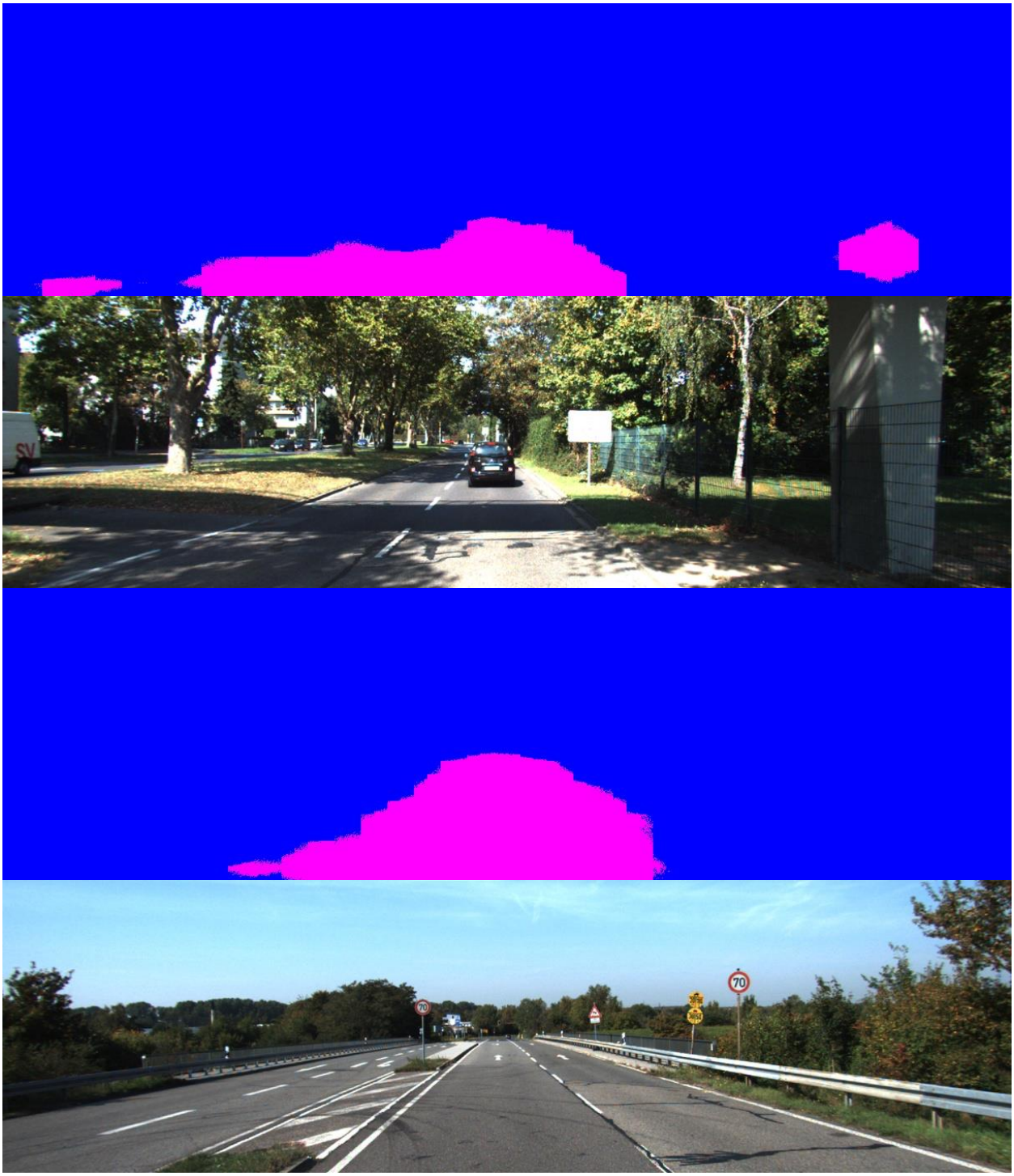


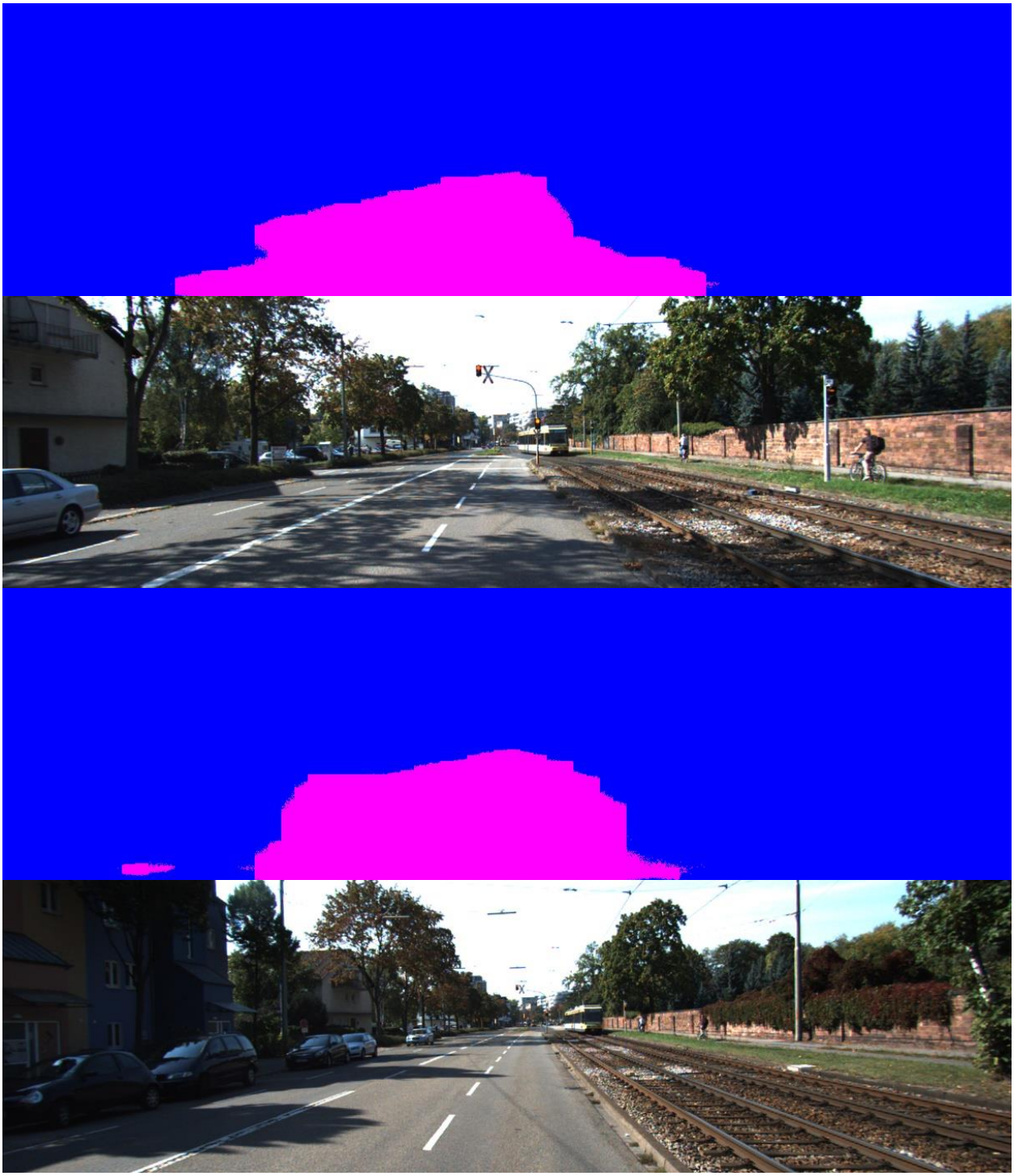
ALDI
Fächerbad

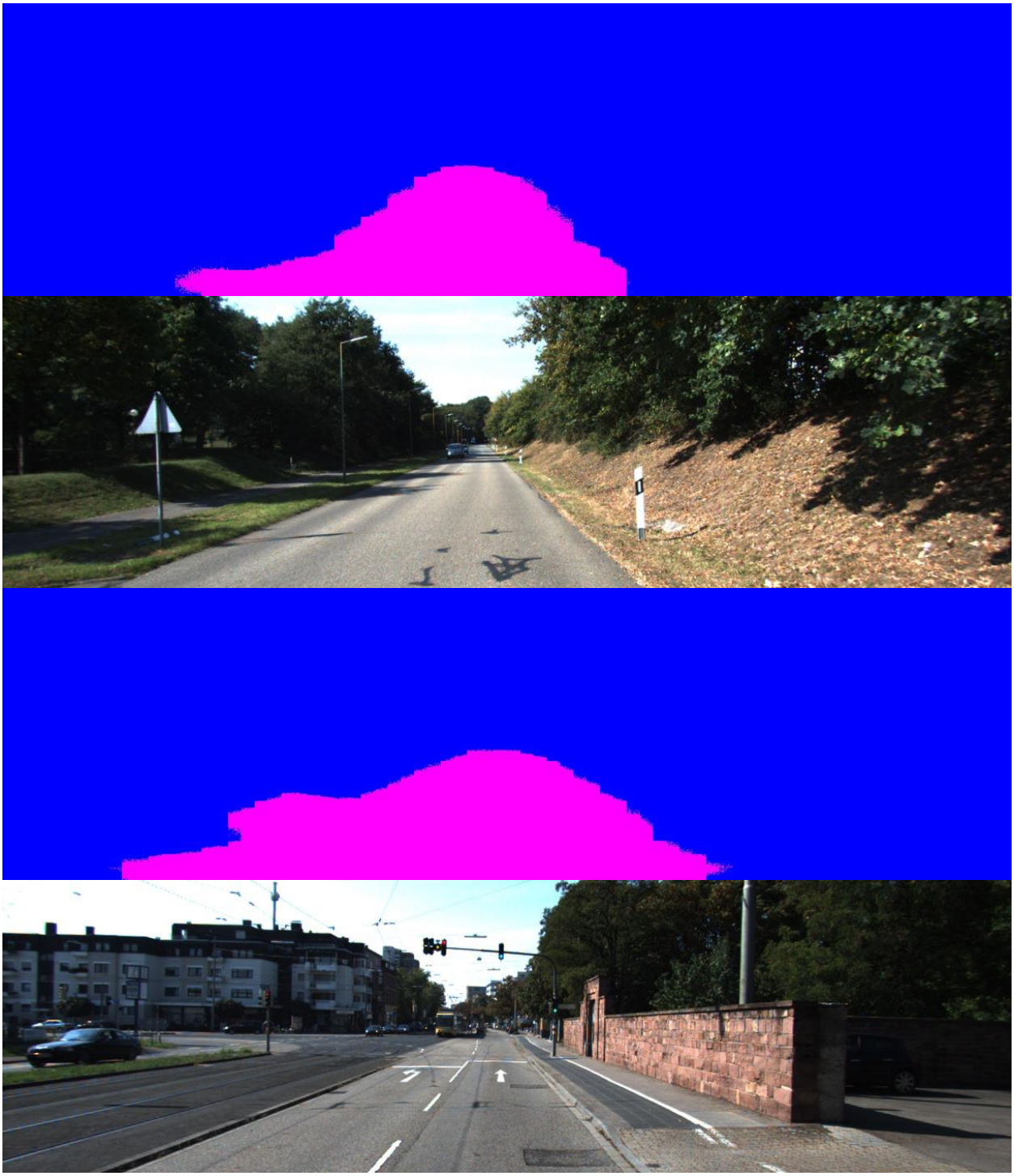


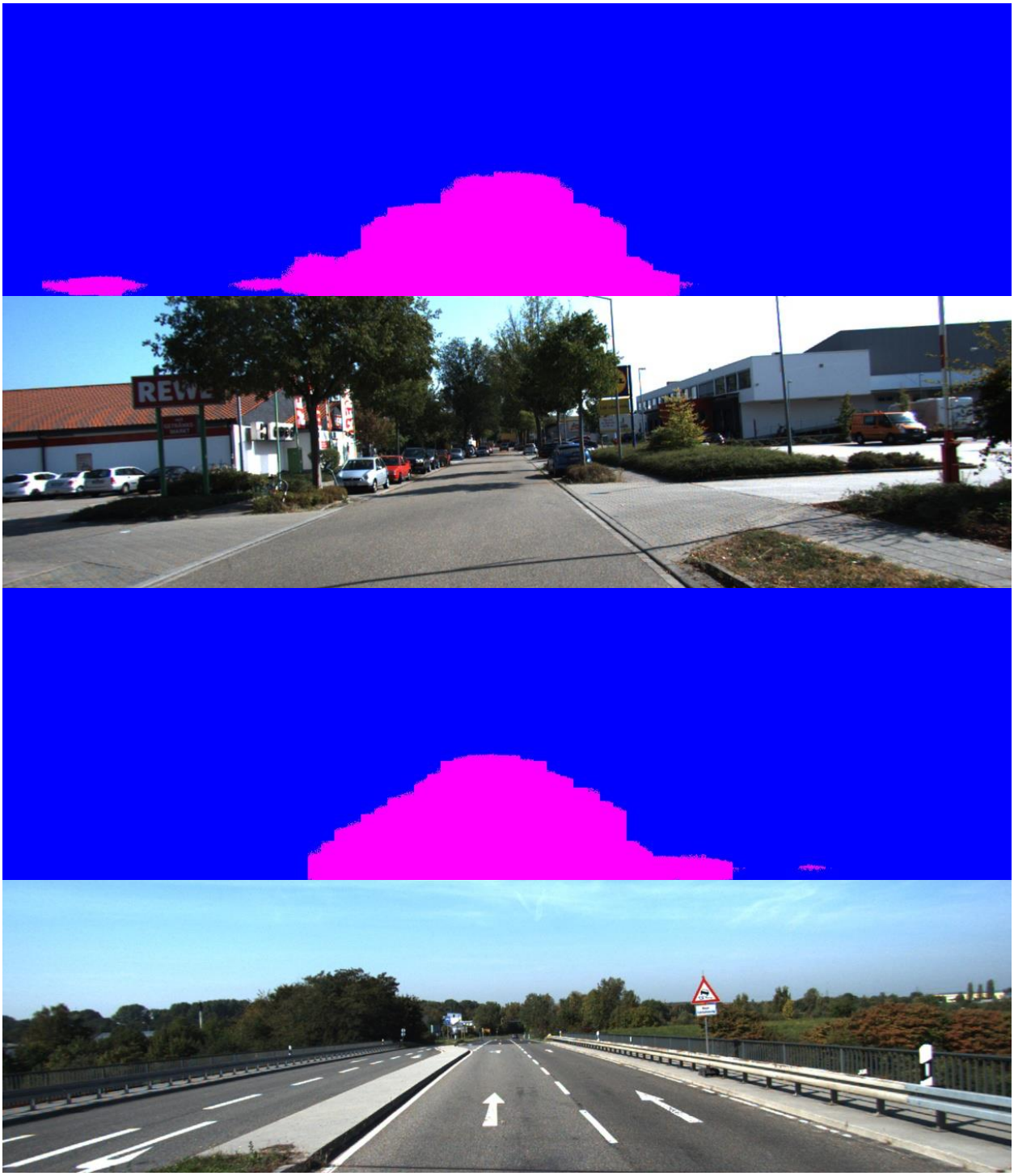

www.seib-natursteine.com



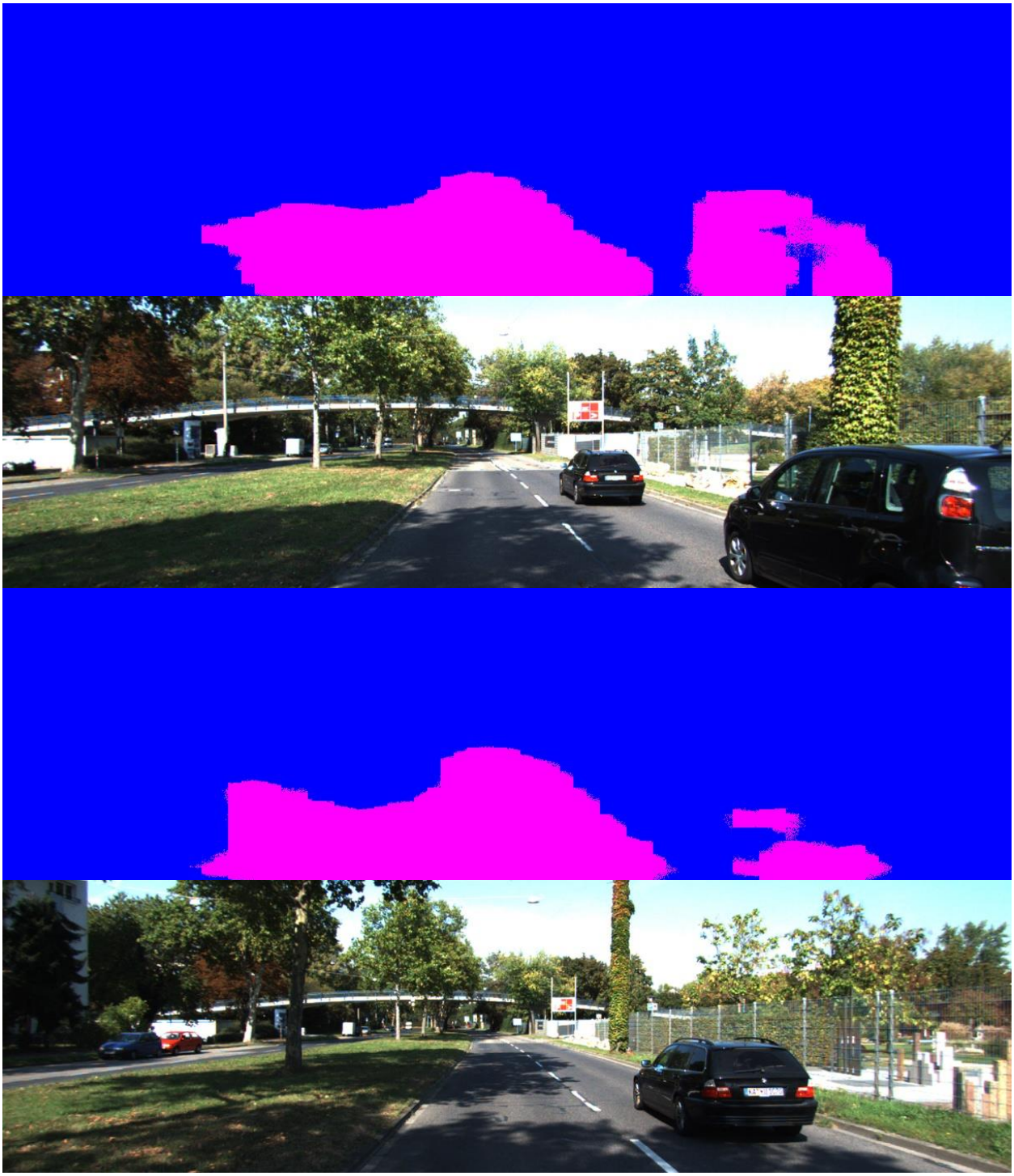

70
70

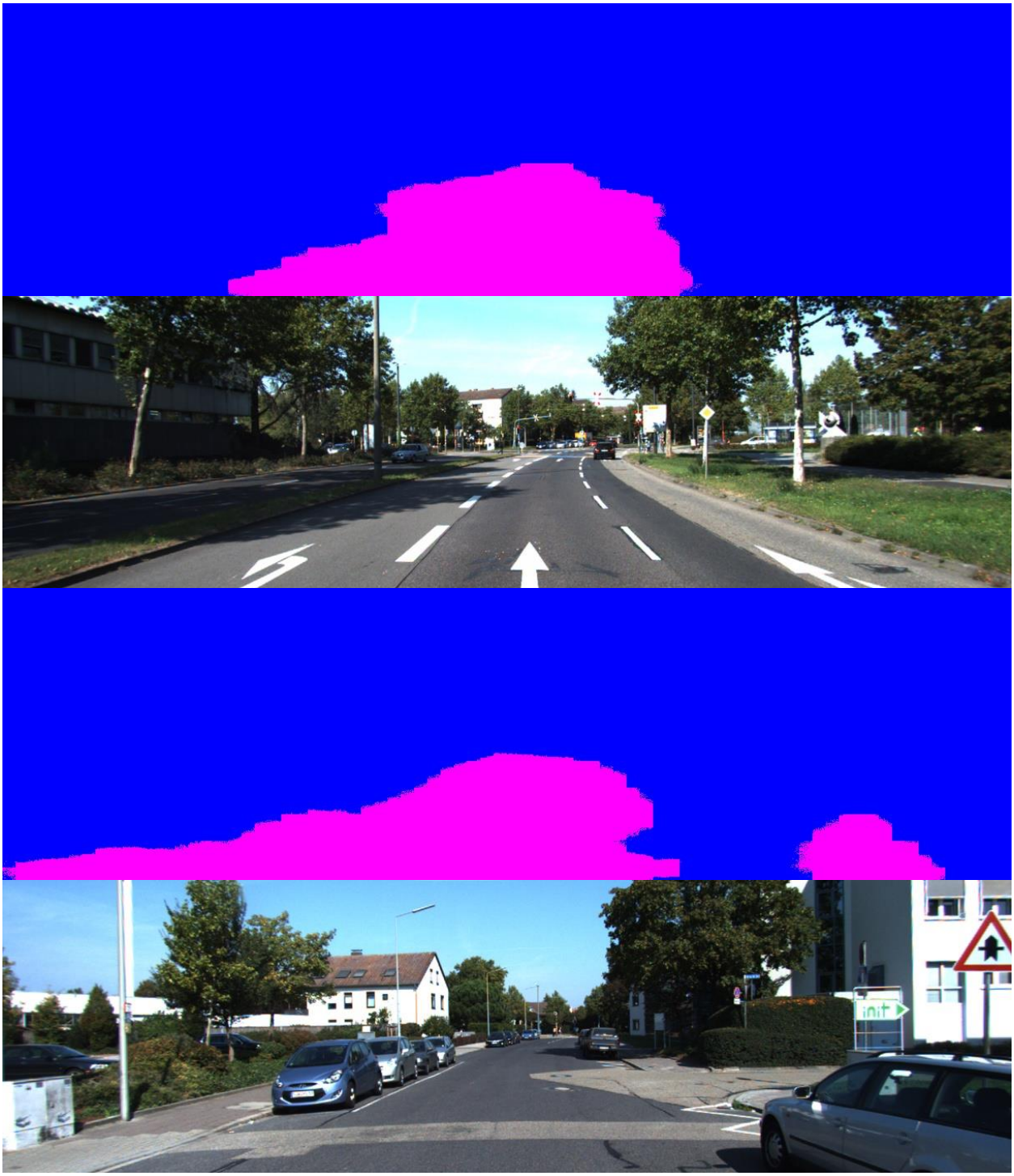


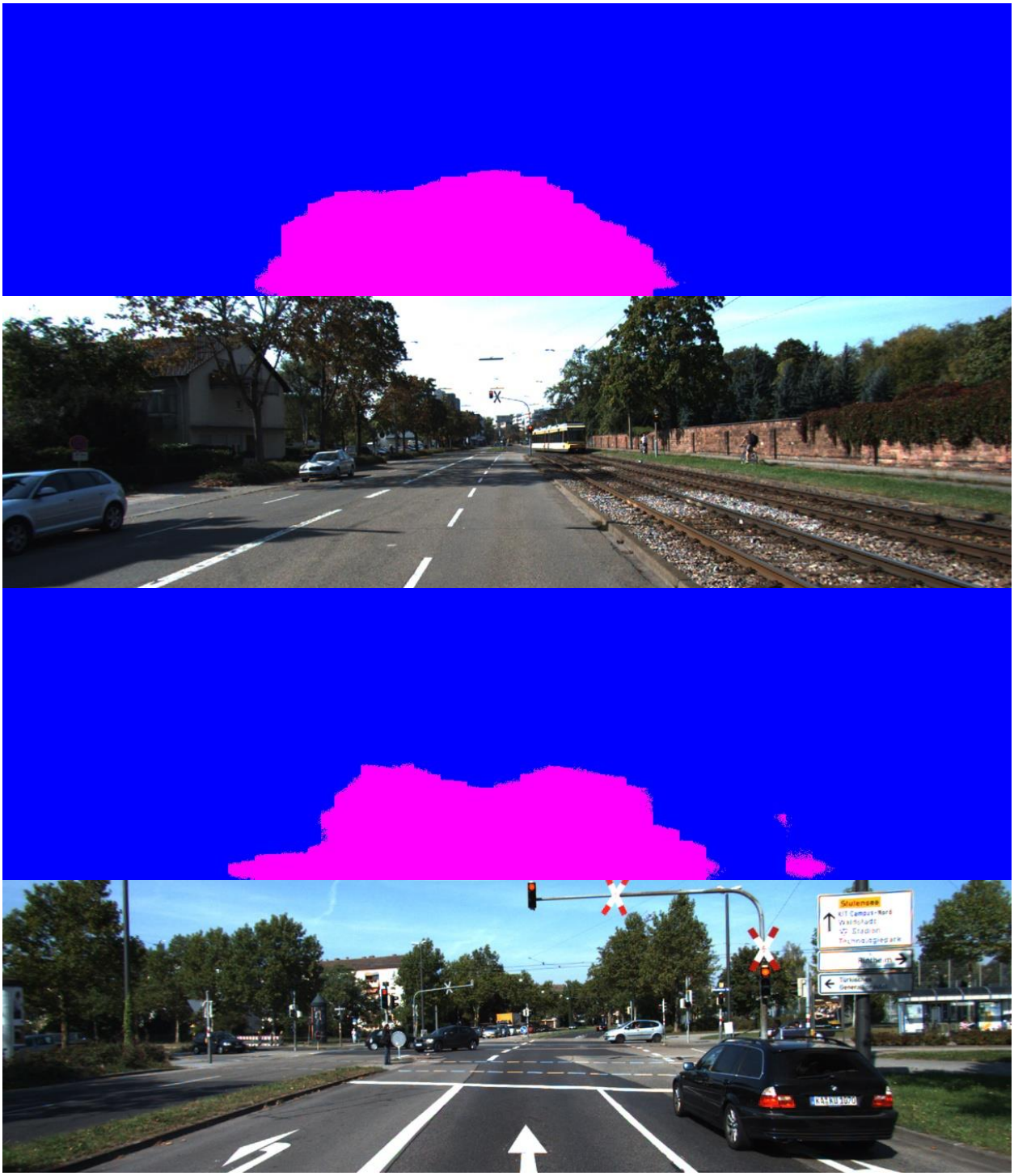


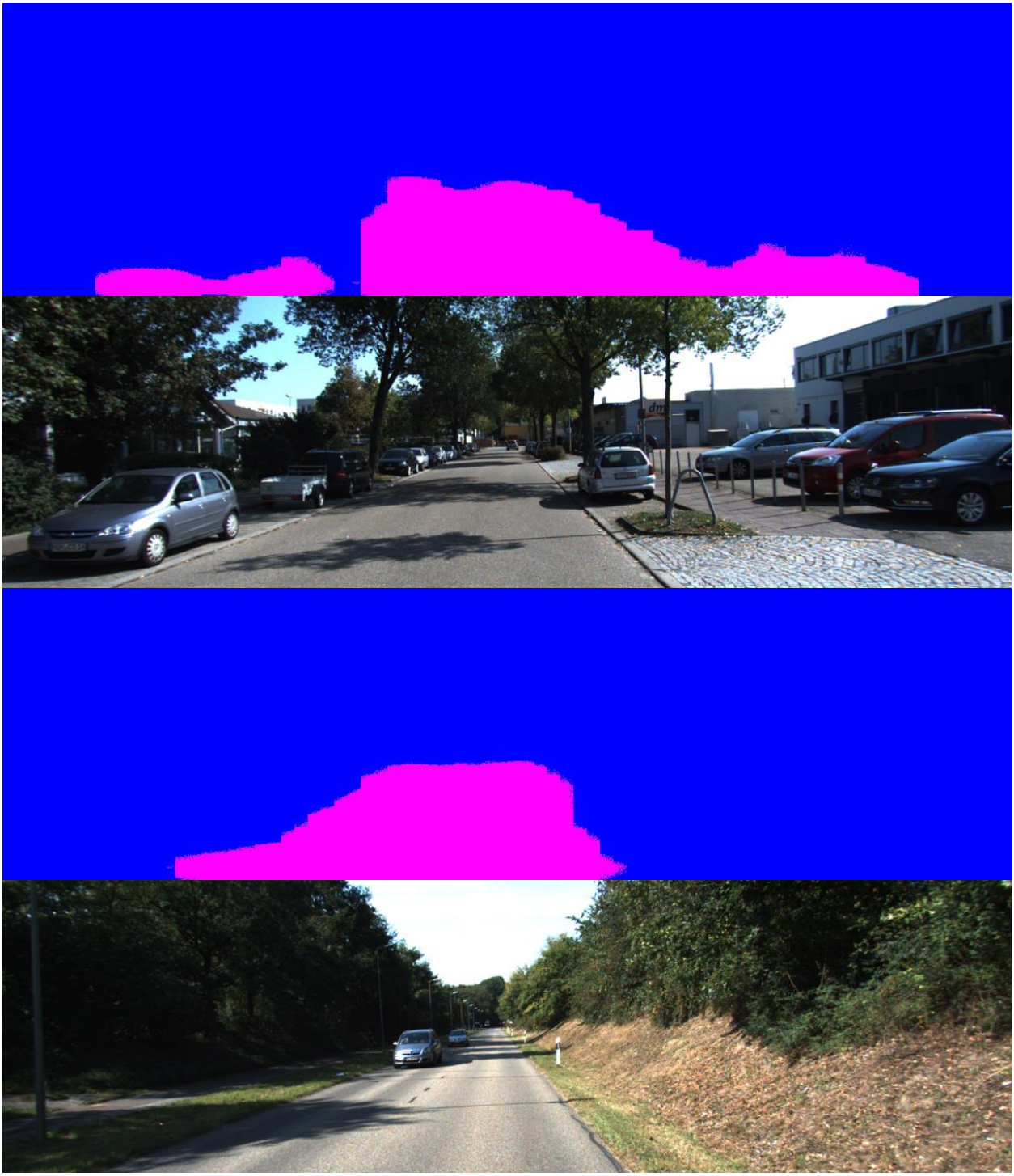

REWE

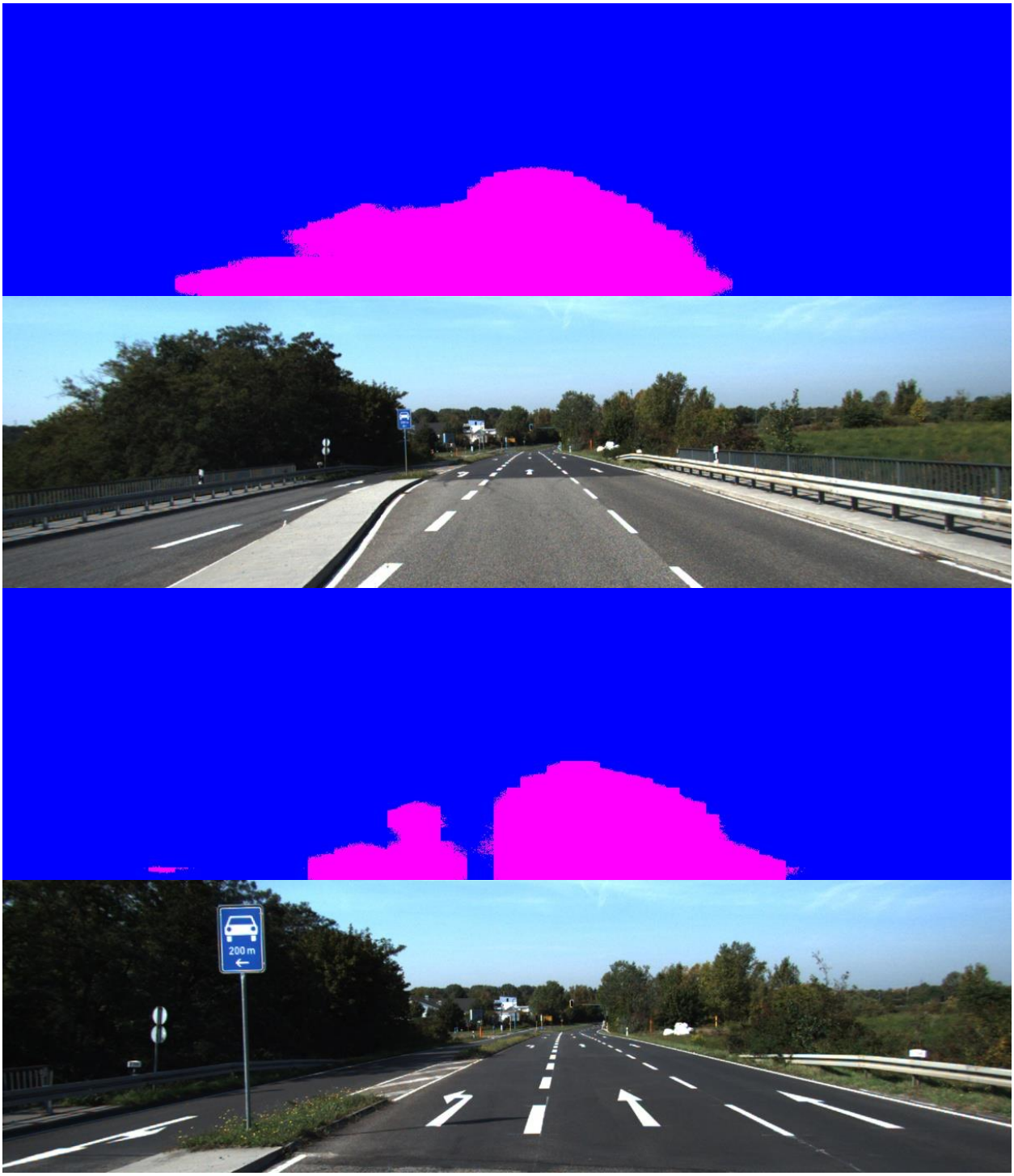
LIDL
Lidl lohnt sich

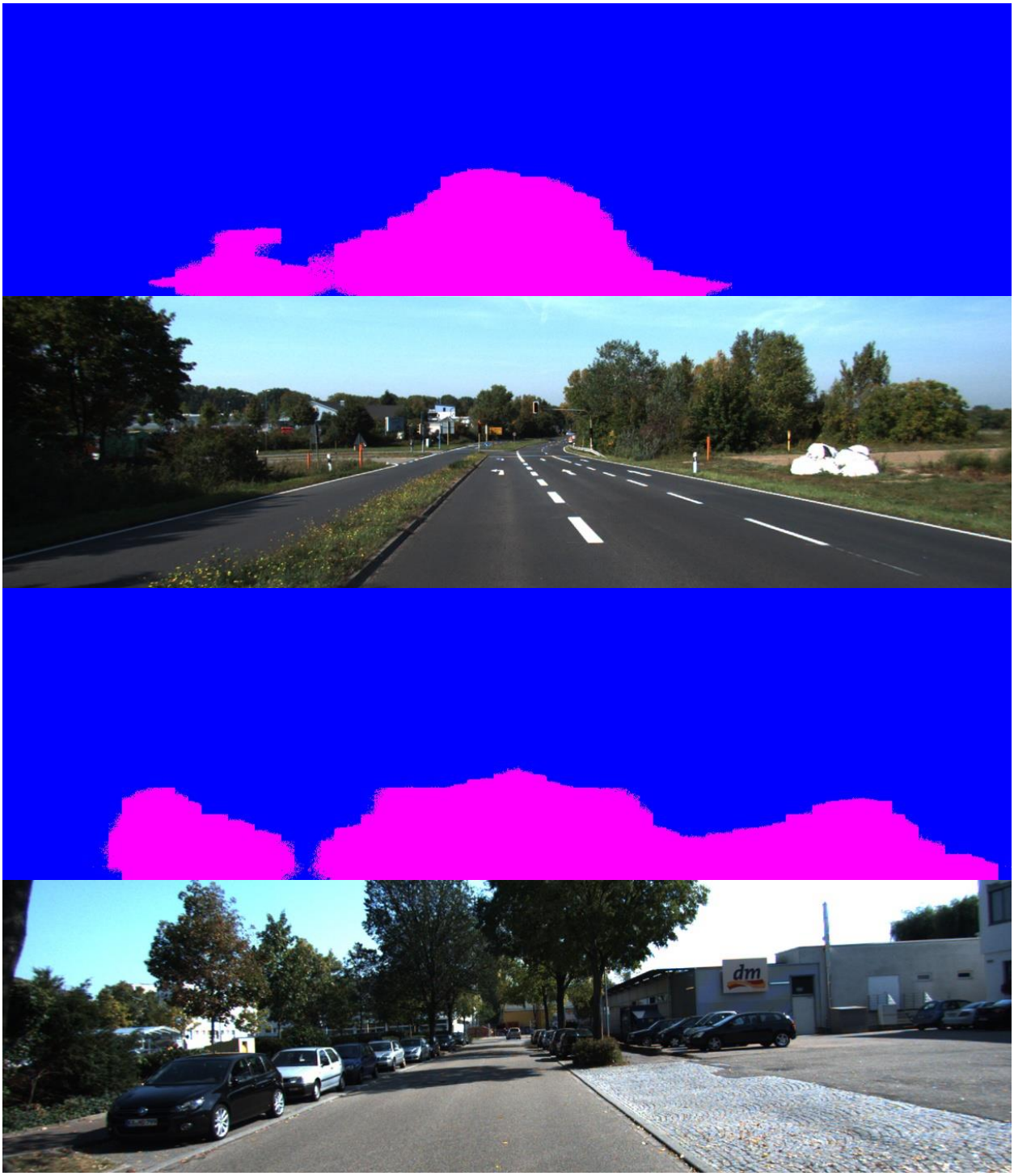
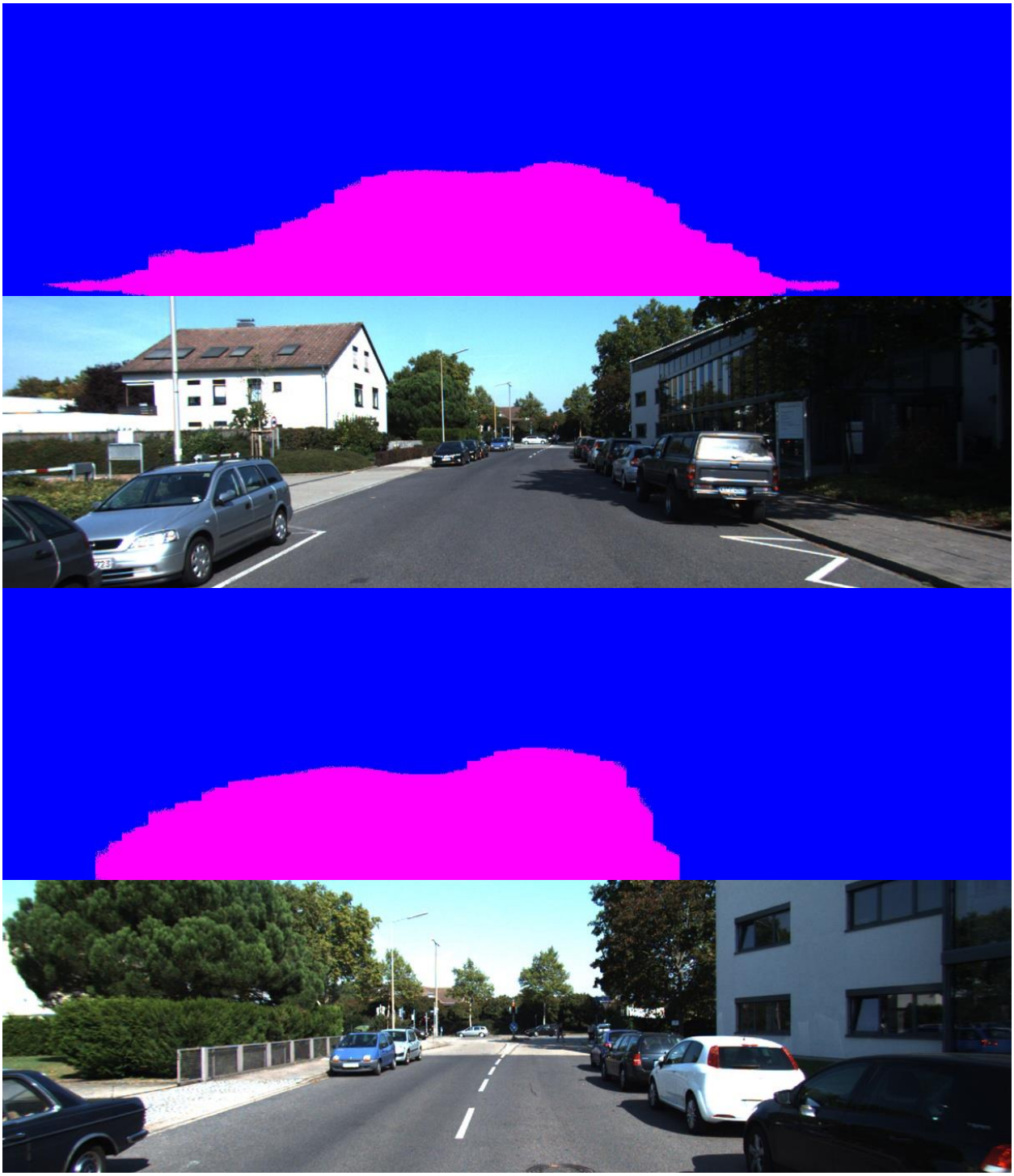
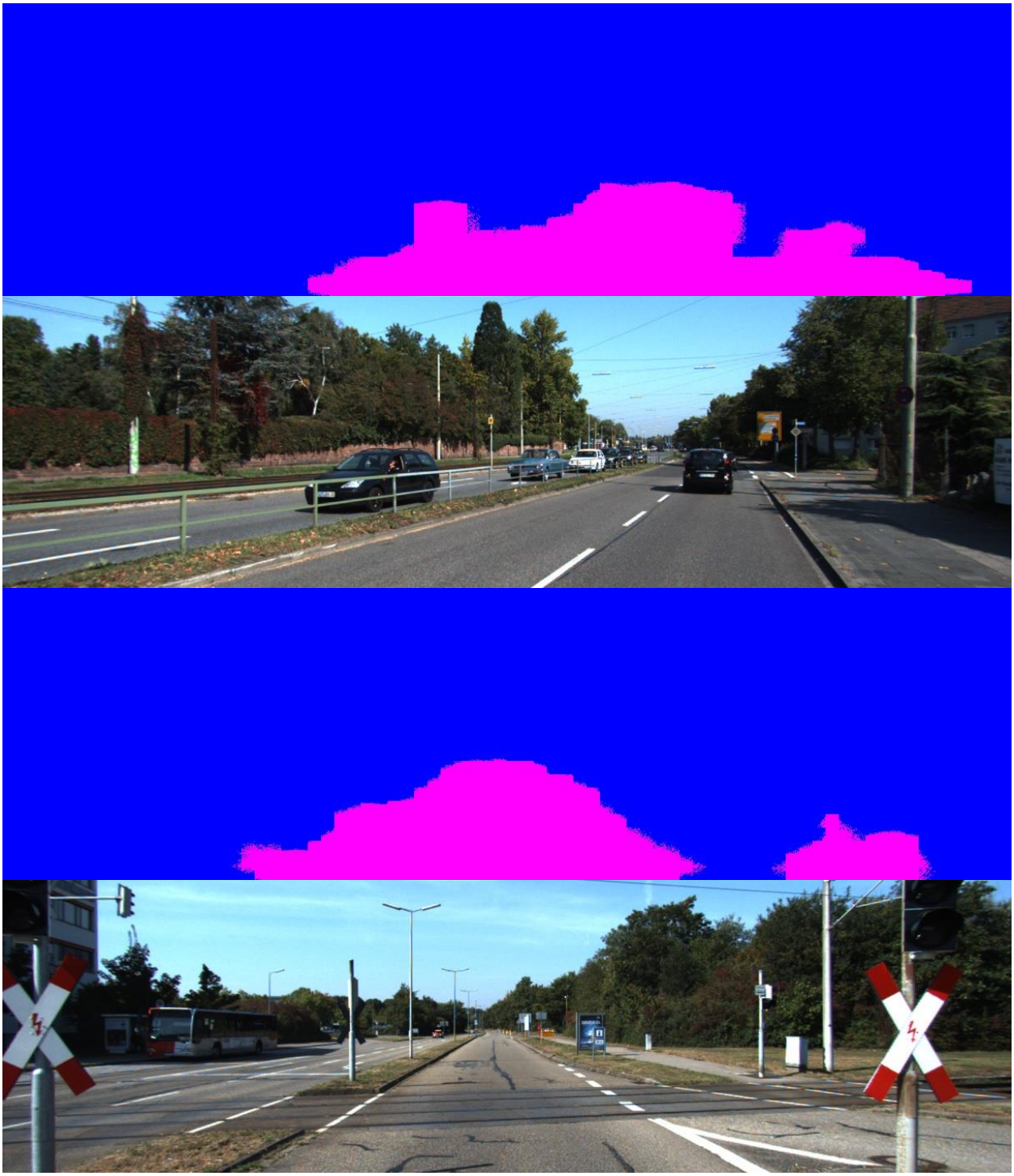
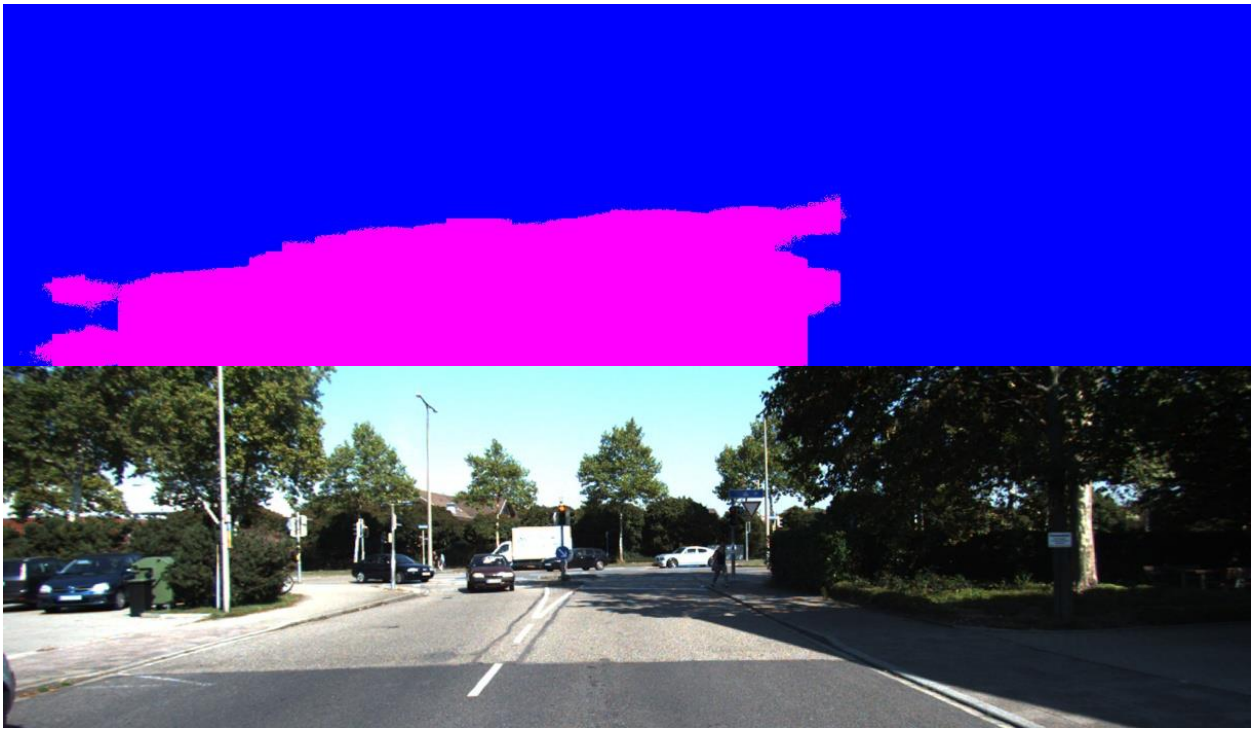



init


Stulensee




200 m


dm