Task 6:

We aim to enhance our navigation system by incorporating image segmentation. While the current system effectively guides us to the destination, there are instances where a BVIP (Blind or Visually Impaired Person) may lose orientation on the sidewalk. To address this, we plan to utilize image segmentation for detecting sidewalks.

Utilizing the Cityscapes Dataset

To facilitate this, we will use the Cityscapes dataset. You can download the Cityscapes dataset from the following link: Cityscapes Dataset

You will need to register and download two files:

1. The original images: leftImg8bit_trainvaltest.zip
2. The annotation images: gtFine_trainvaltest.zip

Use these files to train your model. Additionally, you can find helpful scripts at Cityscapes Scripts, which include tools to convert the gtFine folder into mask images suitable for training.

Training with PyTorch

For training, we recommend using PyTorch. PyTorch has a module with predefined models. Here is an example of how to define a model:

```
import torch

import torchvision.models as models

import torchvision.models.segmentation as segmentation


# Load pre-trained ResNet-34

resnet34 = models.resnet34(pretrained=True)


# Create a DeepLabv3+ model with ResNet-34 as the backbone

deeplab_model = segmentation.deeplabv3_resnet34(pretrained=False,
num_classes=num_classes)
```

Use the number of classes as defined in the Cityscapes dataset.

Defining dataloader for training

For training your model, you will need a data loader to load the left and mask images. Train your model using a data loader like the one shown below:

```
import os

from torchvision import datasets, transforms

from torch.utils.data import DataLoader


# Path to the directories

image_dir = 'path/to/left_camera_images'  # Update this path

mask_dir = 'path/to/gtFine'            # Update this path


# Transformations (if any)

transform = transforms.Compose([

   transforms.ToTensor(),

   # Add any other transformations you need

])


# Custom dataset class

class CustomDataset(datasets.ImageFolder):

   def __init__(self, img_dir, mask_dir, transform=None):

      self.img_dir = img_dir

      self.mask_dir = mask_dir

      self.transform = transform

      self.images = os.listdir(img_dir)
```

```
    def __getitem__(self, index):
        img_path = os.path.join(self.img_dir, self.images[index])
        mask_path = os.path.join(self.mask_dir, self.images[index].replace('.png', '_mask.png'))
# Adjust mask file naming as needed
        image = Image.open(img_path).convert("RGB")
        mask = Image.open(mask_path).convert("L")  # Assuming mask is grayscale

        if self.transform is not None:
            image = self.transform(image)
            mask = self.transform(mask)

        return image, mask

    def __len__(self):
        return len(self.images)

# Create the dataset and data loader
dataset = CustomDataset(image_dir, mask_dir, transform)
data_loader = DataLoader(dataset, batch_size=4, shuffle=True)  # Adjust batch_size and shuffle as needed
```

Please note that these are only examples, and you need to adjust the code accordingly for your use case.

Model Evaluation Using the Mapillary Dataset

After training, download the Mapillary dataset from the following link: Mapillary Dataset From this file, use the test folder to evaluate your model. In the main branch's eval folder, you will find scripts to compute various metrics for your model. First, run an inference from your model, then use the scripts to create binary masks for the sidewalks from your inference images. These images are your predicted outcomes. Also, create ground truth

masks from the original mask images of the 200 test images in the Mapillary dataset. Use the compute IOU script to run different metrics for your model using the predicted and ground truth images. The script expects two folders: one with the predicted binary masks and the other with the ground truth masks. The image names should be the same in both folders. Your model should aim to achieve an IOU of at least 50%.

Important Note

The link to the Mapillary dataset is currently not working. We will update this link on Monday after 12:00. You can start your training without this dataset, as it is only needed for your evaluation.