Task 9:

We now have all the components of the SENSATION system to create an Assistive Navigation System (ANS). Let's build all components of SENSATION together. Check our provided code in the sensation folder in the main branch. Your implementation should be in a file called rsu_vi.py.

Export pytorch model to ONNX

On an edge device like a Jetson Nano, you have limited resources. Therefore, our goal is to optimize the model. Depending on the environment, we don't want to have dependencies on other Python packages. One way to solve this is by converting the PyTorch model to the ONNX format. This way, your application will have dependencies only on the onnxruntime and numpy packages, allowing you to run your model on different devices.

For this, we provide an ONNX exporter. We also provide a class, DeepLabv3Plus_ResNet34. Use these scripts in the helper folder to convert your fine-tuned model. If you used a different model like DeepLabv3Plus, for example, UNet, then adjust the class definition accordingly. Copy your ONNX model into the model_weights folder.

Input handling

For the SENSATION system, we need different input handling. For testing the SENSATION system, we are using images or videos. For live testing we are using the camera input. In the sensation folder, you will find sensation.py and data_handler.py. Use this code parts in your rsu_vi.py script to introduce data handling. Please use the argparser package similar like in sensation.py to have command line options for your rsu_vi.py script. For this implementation your script should provide following commands:

- **Path to an input video.**
- **Path to an GPS coordinate file.**
- **Path to the output video. Where to store the output video.**

Getting environmental information

Use your Valhalla implementation to obtain the route and the corresponding instructions. Use the velocity calculation from the previous task to estimate your

current position. With this, you can estimate the time to trigger the instructions. Additionally, use the segmentation information from your model to identify your localization on the sidewalk. For this, you need to find the dominant column method, which estimates the presence of sidewalk pixels in an image. The method divides the image into three columns: left, center, and right. Then, it estimates in which column the maximum sidewalk pixels are present. For example, if the most sidewalk pixels are present in the right column, you will receive a "go right" command. With this approach, the sidewalk pixels will move to the center column, and the method will give a "stay center" command. It makes no sense to do this for every frame because a video has a speed of 30 frames per second. It is more sensible to check this every two or three seconds. Combine this information with the results from your Valhalla implementation. Introduce a white box in the output frame with the title "Navigation Information" on the left top side. There, make the following visible:

1. **Instruction from VALHALLA**
2. **Command from VALHALLA**
3. **Estimated Speed Current street name Command from segmentation Total time and elapsed time calculated by speed**

Please include in every frame the information even when information is calculated after two or three seconds.

Create video for testing

Create a new video starting from the main rail station in Erlangen and walking to the main university library. Also track the corresponding GPS signals and store them as an XML or JSON file so that your program, rsu_vi.py, can use the video and GPS signals as input. Please do not use your video from the beginning. Create a new video. Look for a route where you walk mostly on the sidewalk. While on the sidewalk, produce a drift to the left and right and then return to the middle of the sidewalk. This way, we can check in the output video if all commands are useful and indicate correct movements. For this, create an input folder in the sensation folder and copy the GPS signals file and the FAU Box link to the original video into a text file. Please do not provide the output video. Only provide the input video. We will run your rsu_vi.py script and check the output.

Implementation of your code

Please try to make your code more modular (define classes) so that we can use parts of your code in our SENSATION system. To improve your code, you can use the Python

library: "ruff." You can use it in two ways: "ruff --fix your_script.py" will fix styling problems in your code. With "ruff format your_script.py," you can format your code to make it more readable.

##Delivery This is the last task for implementation. As discussed in the last meeting, the final date for this task is 29.02.2024 between 18:00 and 20:00, because our next meeting will be on Friday, 01.03.2024. If you have questions regarding this task, please write comments on this task. Good luck.