Assignment 7: Building a Touch Sensor (20P)

Goals

You can apply computer vision concepts to solve non-trivial problems. This way, you can build a DIY camera-based touch sensor. You can detect hand-writing on a custom touch sensor to enable touch-based text input.

1 Build a Camera-Based Touch Sensor 10P

Place the webcam inside the cardboard box facing upwards. Put a sheet of clear acrylic and a diffusor (such as a piece of paper) on top of the box.

Create a new Python program called *touch_input.py*. It should process the camera image so bounding boxes around users' fingers are detected robustly. Make sure there are no false positive detections from a user's palm resting on the surface. Add a calibration step that automatically adjusts thresholds when the program is started, so the system can be used in different lighting conditions.

Your touch sensor should serve as an absolute input device by detecting movement and taps on its surface. You can implement this, for example, by differentiating between a light movement (less pressure/contact on the surface) and a heavier tap (more pressure/contact on the surface). Use DIPPID to send detected events to other programs with the following structure:

```
{"movement": {"x": 130, "y": 200}}, {"tap": 1}
```

Now, it should be compatible with the given Fitts' Law application (*fitts_law.py*), so you can test your touch sensor.

Document design decisions, building process, and usage guide in a markdown file called *documentation.md*.

Score

- (**1P**) Hardware is assembled correctly
- (**3P**) Bounding boxes around fingers are detected robustly, palm-rejection works
- (**3P**) Movement and taps can be distinguished and sent via DIPPID, making it compatible with the given Fitts' Law application
- (**1P**) Automatic calibration works
- (**2P**) Documentation

2 Touch-based Text Input 10P

Find a way to implement hand-writing recognition on your custom touch sensor. Your system should be able to detect text input on your touch sensor entered with a finger or some kind of 'stylus'. In the end, text input should be processed (e.g. letter by letter, word by word, after n seconds, ...) and mapped to keyboard events via *pynput*.

There are several ways to implement this. Besides optical character recognition itself, also remember what you have already learned about computer vision, image classification, or neural networks in previous ITT sessions. Maybe you have to combine time-series data (e.g. multiple images) to one. Maybe you keep the time dimension and work with a model which is designed to handle this. Maybe you can benefit from small applications you have implemented in the past for other assignments. Ideally, your system should not only be compatible with your own handwriting. Keep in mind that some approaches might require a large amount of self-recorded train data – where can you get train data from or how can you circumvent the limitation of missing self-recorded train data?

Document choice and implementation of approach, external sources (if used), and usage guide by extending your markdown file called *documentation.md*.

Score

(**1P**) Detection of finger or stylus works
(**2P**) Input is captured and pre-processed reasonably
(**2P**) Reasonable approach for hand-writing recognition
(**1P**) Robust detection
(**1P**) Acceptable latency
(**1P**) Mapping of handwriting to keyboard
(**2P**) Documentation