Assignment 7: Building a Touch Sensor (20P)

Hand in in groups of two.

Goals

You can apply computer vision concepts to solve non-trivial problems. You can build a DIY camera-based touch sensor. You can implement a simple application with multi-touch support.

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.

As the touch surface has no integrated display, it is hard to predict the position of touch events. Therefore, add functionality to distinguish between low hovering/light touch to preview the position, and strong touch to trigger actual touch events.

Use DIPPID to send detected events to other programs with the following structure (x and y coordinates should be normalized to a value between 0 and 1 with $(0, 0)$ being the top left screen corner):

```
{'events' :
    {0 : {'type' : 'touch/hover',
          'x' : float,
          'y' : float},
     1 : {'type' : 'touch/hover',
          'x' : float,
          'y' : float},
     ...}
}
```

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 and palm-rejection works.
(**1P**) Automatic calibration works.
(**2P**) Robust classification of touch and hover.
(**1P**) Events are correctly sent via DIPPID.
(**2P**) Documentation

2 Multitouch Application 6P

Create a new Python program called *multitouch-demo.py*. The program should load all images in a subdirectory called *"img"*. Those images should be displayed with random size and orientation in a fullscreen pyglet window.

The program should receive DIPPID events from your touch sensor. Process those events so users can:

- move images by dragging them with their finger
- rotate images with a two-finger gesture
- scale images with a two-finger gesture

Score

(**1P**) Images are loaded and displayed correctly.
(**1P**) Images can be moved.
(**1P**) Images can be rotated.
(**1P**) Images can be scaled.
(**2P**) Simultaneous manipulation of multiple images works.

3 Gesture-Based Application Launcher 4P

Create a Python program called *application-launcher.py*. The program should detect at least three different gestures and launch a corresponding program. Gestures should be entered with the touch device or a mouse. The programs should be specified in a plain text file *applications.txt* with one line per program path.

For example, the text file could look like this (on Linux):

```
/usr/bin/xterm
/usr/bin/firefox
/usr/bin/gimp
```

Score

(**1P**) Gesture input works with mouse and touch device.
(**1P**) Three gestures are distinguished robustly.
(**1P**) Applications are launched on gesture input.
(**1P**) Program can be customized with text file and works cross-platform.