EE2703 Assignment 4: Keyboard Analysis

Kaushik G. Iyer (EE23B135)

October 1, 2024

Abstract

In this assignment we were asked to take in a keyboard layout and some sample text. We then had to use this text to analyze our keyboard (Estimate the distance your finger has to travel to type said text and also generate a heatmap showing zones of high usage).

1 Usage

Place all the submitted files `ee23b135_kbda.py`, `ee23b135_layout.py`, `ee23b135_parsing_tools.py`, `ee23b135.py` and `ee23b135.ipynb` in the same directory.

There are many constants defined on the top of each file (the usage of each is defined in the following section and is also specified in the comments) Feel free to play with them :)

In order to run your custom layouts, you may create a json file (similar to the `layout.json` I submitted) or run the code similar to how the last cell in the notebook does (By giving the keys and characters dictionary to `Layout.from_dict`).

You may run either one of the files described below :)

1.1 `ee23b135.py`

Run the file, it will prompt you to give it the path to a valid json file (I have uploaded `layout.json`). You will then be prompted to enter some text to analyze :)

1.2 `ee23b135.ipynb`

Simply run all the cells :).

2 Heatmap generation

NOTE: I don't use matplotlib to render my graphics, instead I use PIL to draw Images :)

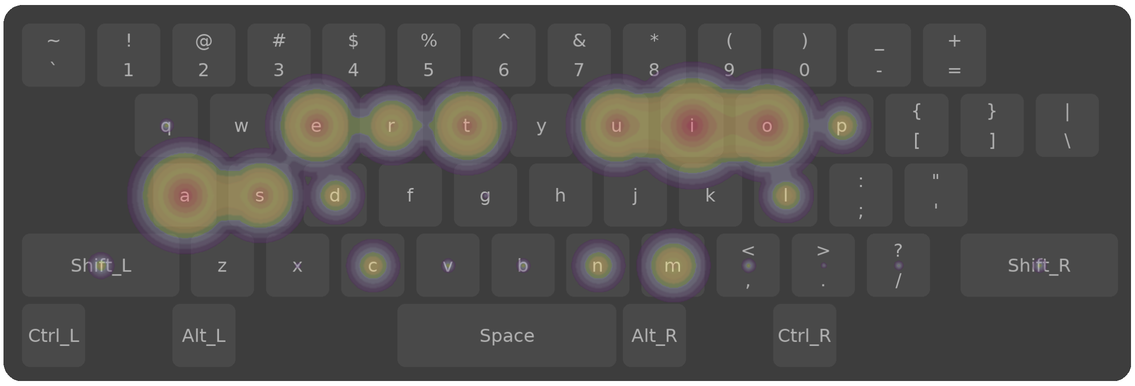

Figure 1: Sample output for lorem50 on QWERTY

2.1 Rendering the keyboard

Refer to `KeyboardAnalyzer._generate_keyboard` in `ee23b135_kbda.py`

1. After parsing the layout I find a mapping from the coordinate space used in the layout to the coordinate space I use on my image. (Based on the constants (like KEYBOARD_SIZE and the KEY_PREFERRED_SIZE related constants) in `ee23b135_kbda.py`)
2. Then I go through every key. If I can I give it its preferred size I do, otherwise its size will be constrained such that it fills how much ever it can while leaving a small padding before the next key. (Again the constants in `ee23b135_kbda.py` are used for this)
3. I also make an effort to properly render keys that have multiple characters on them. (I only do this if DEDUCE_EXTRA_DISPLAYED_LETTERS is set to True in `ee23b135_layout.py`)

2.2 Generating the heatmap

Refer to `KeyboardAnalyzer.generate_heatmap` in `ee23b135_kbda.py`

Realize that there is no need to generate heatmap data on the same resolution as the keyboard. I allow for scaling down (Refer to `HEATMAP_QUALITY` in `ee23b135_kbda.py`) the heatmap data generation (It is then scaled back to the keyboard size at a later stage)

I first generate a greyscale image where the lower the pixel value the higher is the frequency (Initially with everything set to 255). (I took inspiration from heatmap.py)

1. I iterate through a circular region around every key defined by

   ```
   RADIUS = int(
       # The radius must be proportional to the key size (in heatmap space)
       (KEY_PREFERRED_HEIGHT * HEATMAP_QUALITY) *
       # I wanted the heatmap to be more tame for text with less text
       min(total_frequency / 15, 1.5) *
       # A key will have a higher radius of influence the more frequent it was used in
       # proportion to the others
       min(max(0.3, freq * 1.5 / mean_frequency), 1.1)
   )
   ```

2. For every pixel in this region I decrement the pixel value by a ratio given by

   ```
       # The closer the frequency is to max_frequency, the smaller addition is done to dist
       # Smaller addition to dist implies that the ratio (pixVal) will be smaller
       #(ie higher value on heatmap)
       reduction_factor = (dist + FUZZ * min(max_frequency / freq - 1, 1) / 4) / FUZZ

       # Some extra dependancy on the frequency (wrt total frequency this time)
       reduction_factor *= max(total_frequency / (freq), 1)**0.2

       pixel *= reduction_factor
   ```

3. **Contouring**: Now the image is very smooth, to get contour lines (similar to the example website). I make all values with the same quotient when divided by `HEATMAP_CONTOURING` (A constant defined in `ee23b135_kbda.py`) to the same value.

   `Try setting HEATMAP_CONTOURING to 8 to get very smooth bubbles :)`

4. After this I invert the greyscale image (thus making high frequency correspond to higher values), resize it to the required image dimensions and run a gaussian filter over it (to smoothen it a little).

2.3 Converting greyscale to color

I've attached some samples where I don't colorize my maps (so you get an idea of how the greyscale image looks).

1. In order to color the grayscale image, I define 3 groups of pixels high, medium and low (Defined by their greyscale values).
2. I then find the normalized values of each pixel in a group (Make all between 0 to 1 based on their value in their group)
3. Then for each group I linearly interpolate the pixel's color between 2 predefined colors (different for each group) based on this normalized value.

Once this is done the colored heatmap and keyboard image can then be composited together to create the final image :)

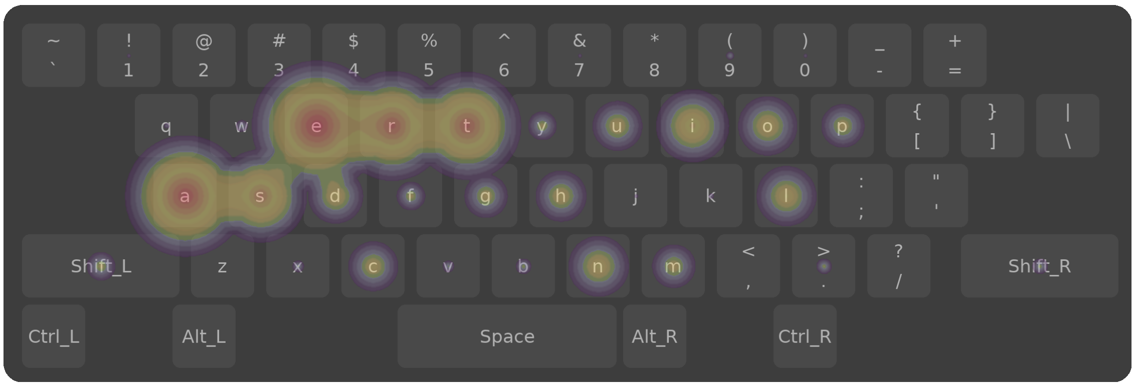

Figure 2: Sample output for sample4 (from here) on QWERTY

3 Finger movement calculation

I've allowed for 2 different types of distance calculation. Toggle `BASIC_FINGER_MOVEMENT_CALCULATION` in `ee23b135_kbda.py` to choose which to use :)

3.1 Basic Model

Whenever a key is typed, the finger responsible for it (defined by in the layout) moves some distance from the start key (also defined by the layout) to the key to be typed. It then teleports back to the start key with 0 extra distance added. (As described in the assignment)

1. Distance calculated on lorem50 was 372.1751451556409
2. Distance calculated on sample4 was 711.7114244494628

3.2 A ~~better~~ Model

Whenever a key is typed, the finger responsible for it (defined by in the layout) moves some distance from the key it is currently on (It starts initially at the home key) to the key to be typed. It then **remains** at that key till it is required to type another letter that it is responsible for. :)

1. Distance calculated on lorem50 was 428.7359446654423
2. Distance calculated on sample4 was 866.3901248345229

4 Comments

Refer to `ee32b135_sample_text` for `lorem50` and `sample4` texts.