Computer Vision

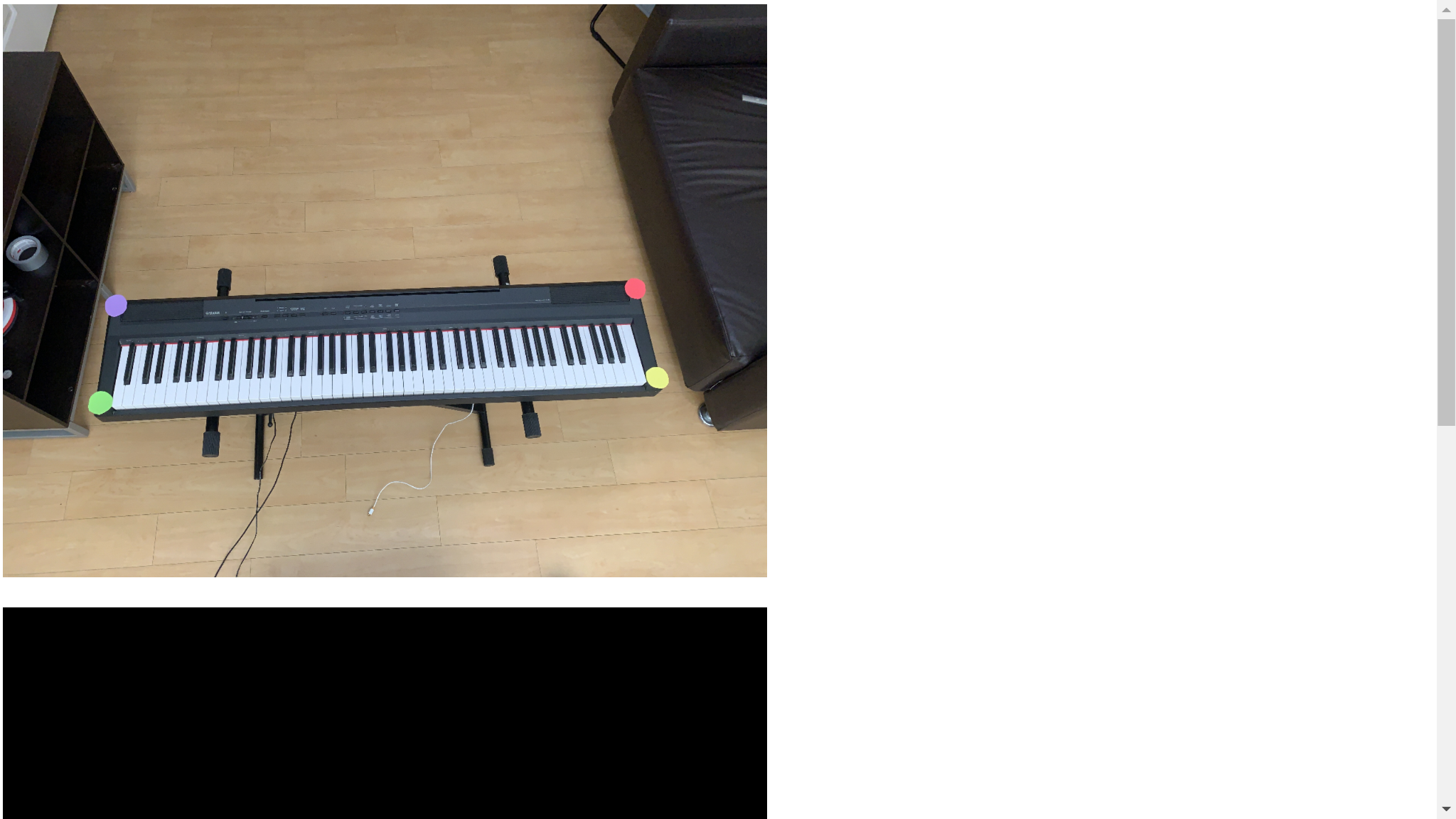
Semester Project Part 1 Deliverable

Daniel McCarthy and Chris Hunt

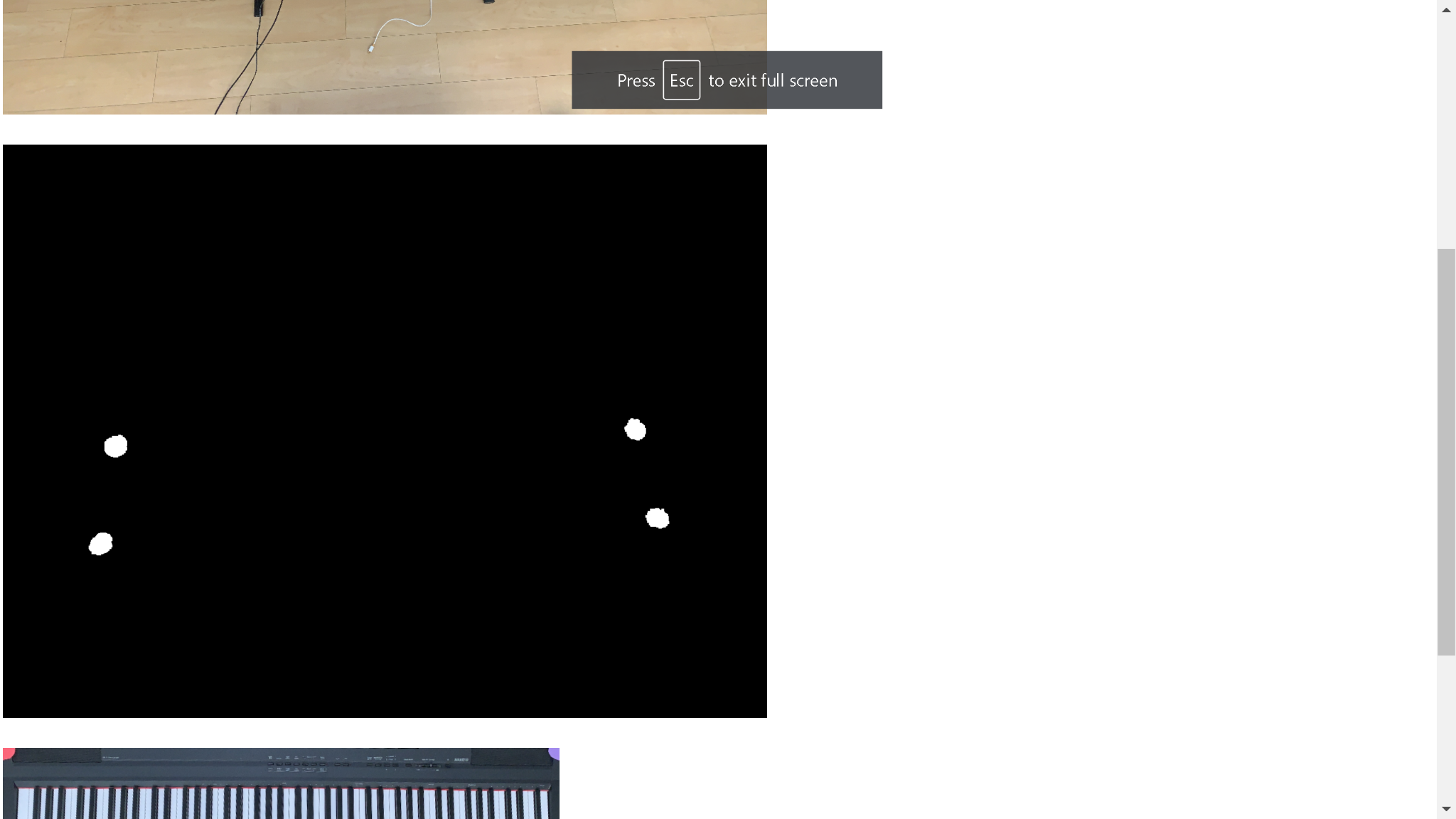
Data preprocessing, segmentation, and feature extraction

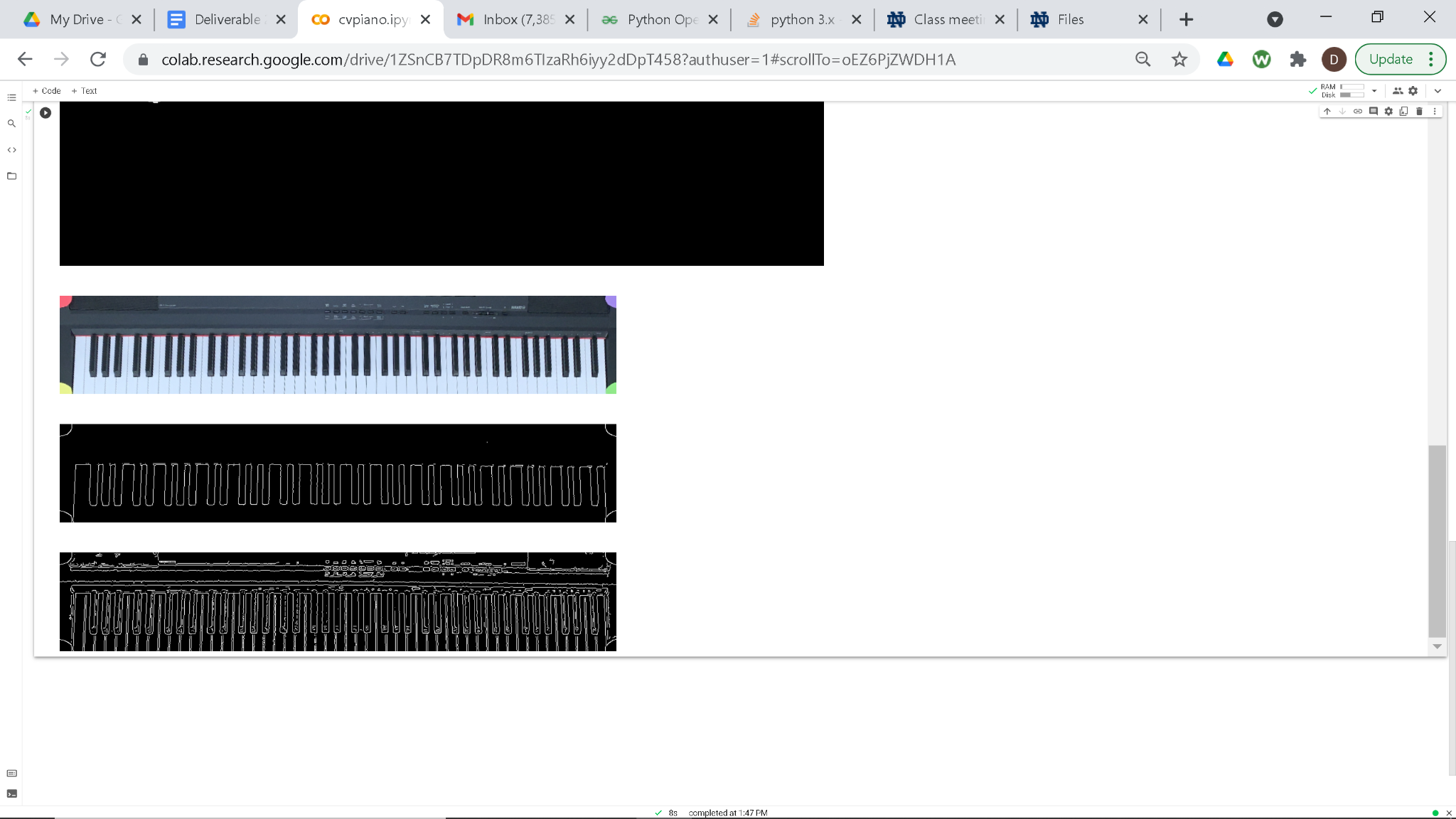
1. A list of the methods already applied for data pre-processing and feature extraction **(2 points).**
   1. Use HSV selection and Morphological operations to locate colored stickers.
   2. Can use measure.label and measure.regionprops on our extracted stickers to locate positions of stickers.
   3. forwardWarping to straighten the piano image according to the position of our stickers.
   4. Edge detection/Hough transform to detect lines between keys and locate front/back of keys to isolate keys from the rest of piano/background.
2. A short justification why you decided to use these algorithms **(3 points)**. For instance, if you used Canny edge detection and Hough transform to detect lines, say why you believe this feature extraction is good for your project.

The images were taken with 4 different colored stickers located on the corners of the piano. Using HSV selection to determine the range of each color along with morphological operations allows us to efficiently detect the corners of the piano via a mask. We then take the sum of each color corner’s mask to create a new mask which detects all 4 corners simultaneously. This results in a binary image with separate unconnected pixels marking the position of each corner. Now, Skimage library contains the function measure which has a method to return coordinates of detected areas. We can use this function on our mask to effectively detect the exact position of our piano in any image with any orientation. These coordinates will be used to segment the piano from the rest of the image. Now, we implement forwardWarping to straighten our image. We want the final image that is passed through the function that detects piano notes being played to have the same orientation as all other images. forwardWarping can do this for us. Finally, Edge detection involves a kernel with a high center value and low surrounding values. This allows detection of stark changes between pixel values. Edge detection is great for small features, so it would allow us to get an outline of the keys and separate them accordingly.

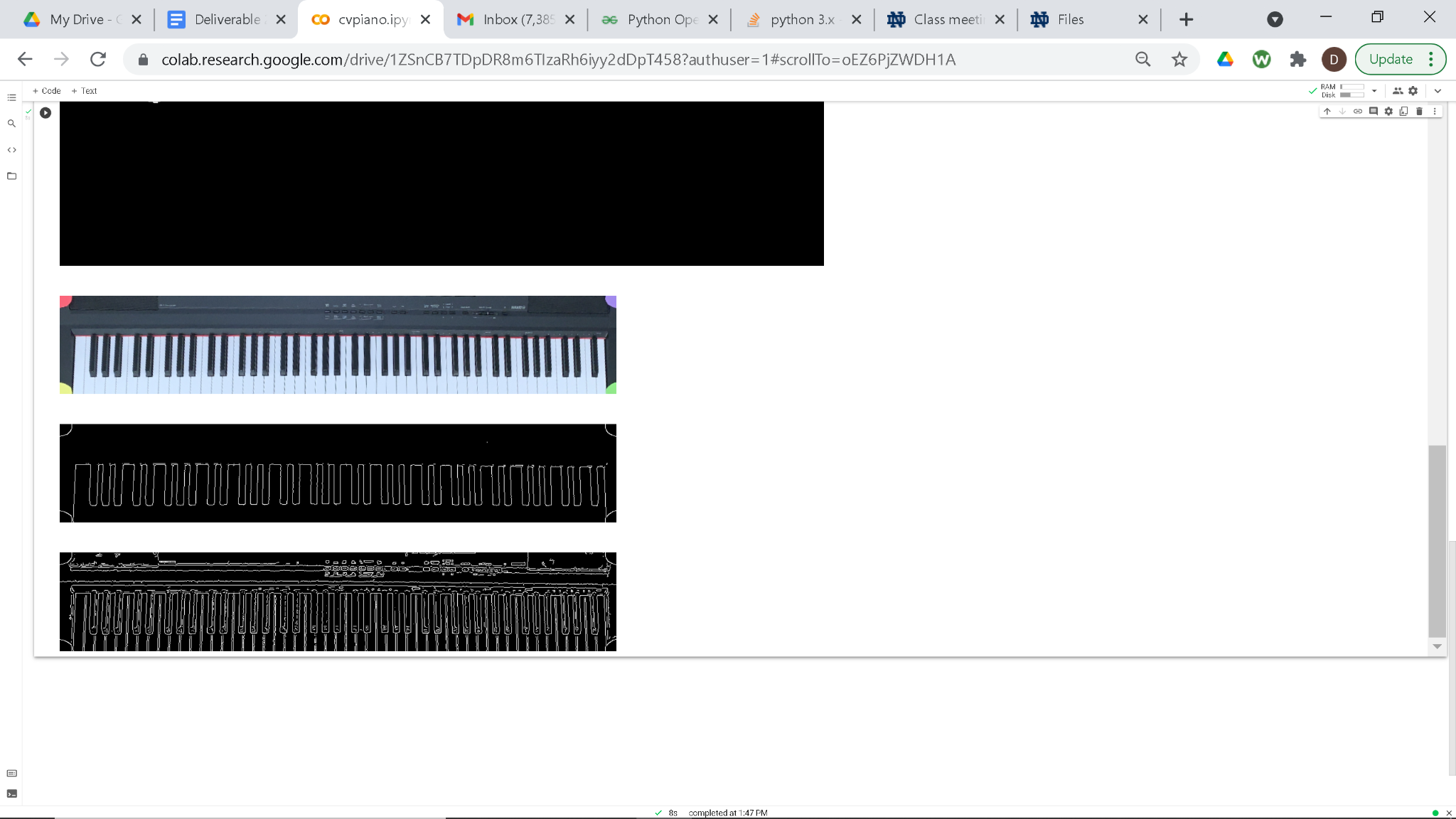
1. A few illustrations demonstrating how your methods processed training data, for instance segmentation results **(2 points).** 

This first image is a sample picture of our dataset, showcasing the piano with stickers marking each

of its corners.

After masking each of the colors of the stickers, our resulting mask looks as shown above.

We now use the position of these white pixels to both crop and straighten our piano image, focusing in on the actual keyboard.



Finally we apply Canny edge detection to first detect black keys, and then all keys, as shown above.

Github:

<https://github.com/chunt4/cv-piano>

cvpiano.ipynb:

<https://colab.research.google.com/drive/1ZSnCB7TDpDR8m6TIzaRh6iyy2dDpT458>