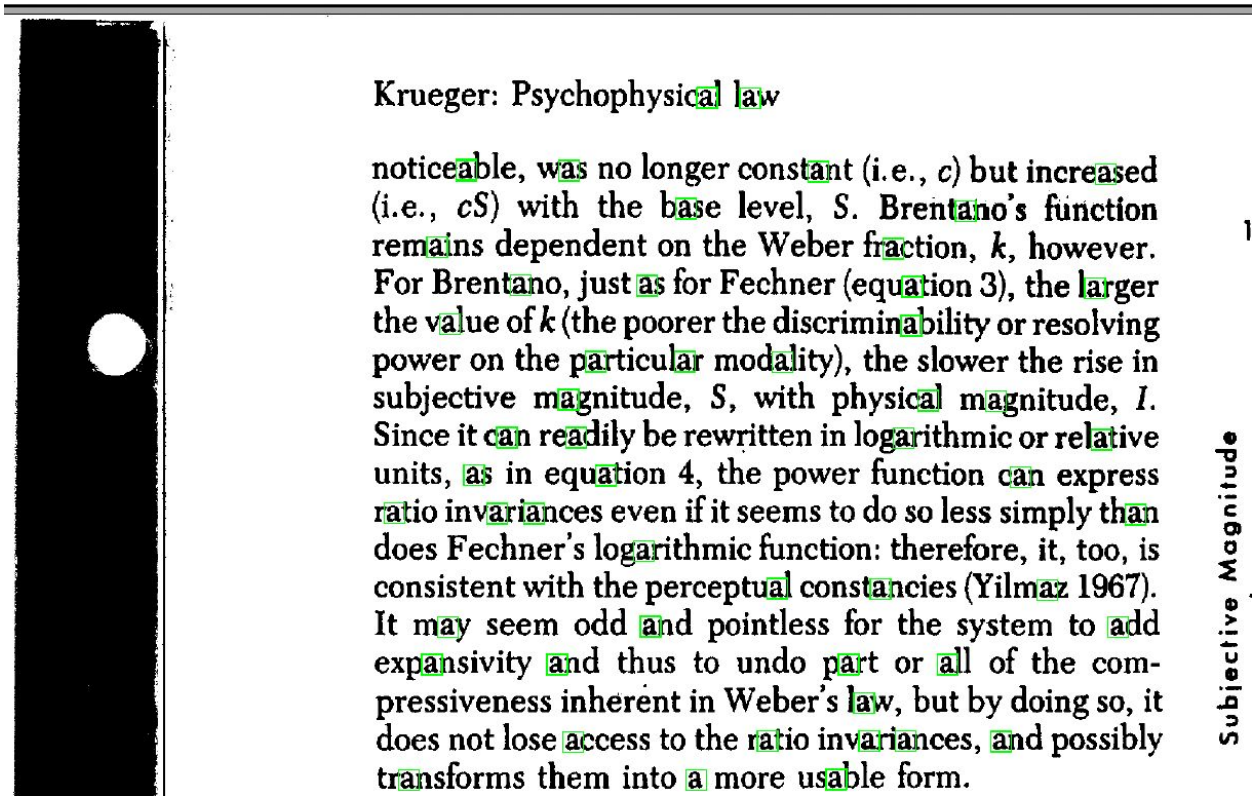
2)
The approach to this problem was pretty straight forward, I began by leveraging past code to allow for template submission that would log the pixel coordinates of the template, this allowed me to easily find a template that could give me the best results. Once template coordinates were found the callback was commented out and the template coordinates were hard coded in. I then ran this template through OpenCV's match template function producing a matrix of correlation scores. I then created a subset matrix using np.where which gave a matrix with elements that matched my conditional, in this case any correlation score higher than an experimentally derived threshold score. Once this was found I could simply iterate over the constructed matches matrix and make a bounding rectangle over all the elements.

Template:

Output:



Number of a's: 104

4)
The method I took required a bit of fiddling. I started by converting each frame of the video into a binary image using the OTSU method and a complimentary inverse binary image. I then found all the white blobs first using the connectecComponentsWithStats method to also give the centroids information. This was then repeated to find the black centroids. I was then able to do a

n^2 comparison of all the blobs singling out any where the euclidian distance was less then the found threshold, the white blob area was less than the black blob area and the black blob was completely contained in the white blob. At which point this can be considered a CCC.

Video Link:
https://youtu.be/hqe0376FpJs