Thresholding

The first algorithm I used was the default cv2.threshold with the default parameters provided by the example code. This resulted in almost everything except the computer screen and parts of the upper half of the image A being blacked out as part of the background. When I reduced the threshold parameter from 200 to 150, I returned much better results. Less of the image was blacked out as part of the background while the computer screen successfully passed above the threshold (as expected).

The second algorithm I used was the cv2.adaptiveThreshold method. I suspected this would produce superior results due to it determining a threshold for each pixel based on its surrounding neighbors. Initially, I had the threshold value calculated by taking a simple mean of the given pixel's neighborhood using the adaptive_thresh_mean_c parameter. This produced significantly superior results to the simple thresholding as only a minimal amount of background fell below the threshold. Using a Gaussian-weighted sum of the pixel's neighborhood (adaptive_thresh_mean_c) further improved the adaptive thresholding results. In all adaptive thresholding cases I found a 15 block size and subtractive constant of 9 to produce the least noisy results.

Filtering

An 8x8 kernel size appears to work well for the *A_original_image.jpg* but not for the *1_ms_surface.jpg* image when applying morphological opening. The 8x8 kernel results in parts of the laptop screen being filtered out which is not what we want. In comparison it has no effect on the laptop screen for A_original_image.jpg. Accordingly, we require a smaller kernel size for the larger *1_ms_surface.jpg*; a 2x2 seems to work better.

However, the 2x2 kernel size works well for *1_ms_surface.jpg* if we apply morphological closing immediately after. I found that 10-14 iterations of morphological closing helped remove almost all noise while ensuring the laptop screen remained unaffected. Some signal is lost at the bottom portion of the screen, but more signal is regained where the icons on the screen was previously removed. Only 2 rounds of closing is sufficient on *A_original_image.jpg* to remove noise while keeping the laptop screen untouched.

Connected components

After calculating the area of each component, my first idea was to filter out all areas smaller than a threshold and areas larger than a threshold, so that only those in between would pass. This worked well for *A_original_image.jpg* but I quickly realized it would fail for *1_ms_surface.jpg* due to the much larger screen size. Therefore, I switched to using aspect ratios rather than areas to filter out any components with a ratio larger than 1.5, as laptop screens are generally rectangular. This worked significantly better.

Contour properties

In addition to another round of aspect ratio filtering, I determined that the extent could be used to filter for screens. For this I kept it simple and just used 1.0 as the lower bound. At this point, only a few false positives remained, which I managed to screen out by selecting only the largest contour for passing. This appeared to work well for *A-orginal_image.jpg* and somewhat for *1_ms_surface.jpg*. It worked exceptionally well for *5_screen_in_bkgd.png* except it could not recognize the second screen in the background because my filtering method only allowed for the