I stuck with HSV as the base approach since that worked best before, but I expanded the color ranges and added some heavier morphological operations (closing + larger kernels) to try to smooth out gradients and fill gaps. The idea was to make the shapes more solid and less sensitive to background patterns bleeding in.

I also experimented with a couple of other ideas while working on this. One was pixel-by-pixel detection, where I tried blending across gradients to get cleaner masks. That technically worked in some cases, but it was super computationally expensive and still didn't handle transitions very efficiently. Another big challenge came up with shapes that were close in tone to the background — especially the white/gray/black shapes. Since the background and those shapes overlap a lot in HSV space, it was hard to separate them cleanly. On top of that, colors like purple and green were giving issues because their RGB values were so different even though they looked similar visually, so the ranges were tricky to tune.

Overall, the current algorithm can now detect part of the white & black shape (doesn't fully trace the black part of it), and it can handle the red-yellow gradient (because it has similar RGB values). However, I was unable to make it function for the shapes with rougher gradients.