Part Three: Reducing memory access across kernel-user boundary

Implementation

My implementation has a local server rust file and a remote server rust file – the local server captures the image from the video stream in the App struct, and then sends that image over a TCP stream to the remote server using the ServerFacing struct. The remote server reads the image from the stream, turns it into RGB, applies the model to the image, and then sends back the keypoints and RGB image as the result. The local server will then convert the RGB image to BGR and render the keypoints as well as the image on the screen.

Difficulties

I had a lot of difficulties understanding the conversions between types – for a long time I had a blue image on my screen because imshow requires a BGR image instead of an RGB image, so I had to do that conversion as well. Understanding the size of the image as well was difficult as was understanding how to flatten and resize the image and when to do those things.

AI

AI wrote most of my baseline code – it gave me the RGB conversion which was nice. It also helped me understand the image to vector conversions and how to send the image + the keypoints back to the local server.