CS 5330 Project1
Yunyu Guo
Jan 19 2025

This project is to apply filters to live video, capture, manipulate and write images. Also use the Depth Anything V2 Deep network for estimating depth values from images. Utilize the face detection and toggle keys to switch effects. Filters include single-step pixel-wise modification and area computation (like a Sobel or blur filter).

- **3. How each color channel is weighted in the conversion**
  RGB[A] to Gray:$Y \leftarrow 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B$
- **Required Image 1: show the original and cvtColor version of the greyscale image in your report.**

- **4. How you decided to generate your greyscale version, and highlight the differences with the default greyscale image.**

  Use custom grayscale formula:
  int gray_value = (255 - red) * 0.4 +green * 0.5 +blue * 0.1
  Compare this to OpenCV's standard grayscale formula, the custom formula inverted the red channel, reducing the influence of green and blue.

  Also the custom grayscale method used the same gray_value for all channels to create grayscale effect. This is different from a single-channel grayscale image (OpenCV's cvtColor) because the custom grayscale method maintained the 3-channel structure while making it appear grayscale.

- **Required image 2: show your customized greyscale image in your report**

- **5. Show your sepia tone filter in your report and explain how you ensured using the original RGB values in the computation.**

  ```
  // Sepia coefficients
    const float sepia_r[] = {0.393f, 0.769f, 0.189f};  // Red coefficients
    const float sepia_g[] = {0.349f, 0.686f, 0.168f};  // Green coefficients
    const float sepia_b[] = {0.272f, 0.534f, 0.131f};  // Blue coefficients
  ```

```
// Calculate new values using original sepia coefficients
        int newBlue = std::min(255, (int)(blue * sepia_b[2] + // B * 0.131
                                          green * sepia_b[1] + // G * 0.534
                                          red * sepia_b[0])); // R * 0.272

        int newGreen = std::min(255, (int)(blue * sepia_g[2] +
                                           green * sepia_g[1] +
                                           red * sepia_g[0]));

        int newRed = std::min(255, (int)(blue * sepia_r[2] +
                                         green * sepia_r[1] +
                                         red * sepia_r[0]));
```

- **Required image 3: show your sepia tone filter in your report**

Extension: **add vignetting (the image getting darker towards the edges) to this filter.**
Vignette is defined as : float vignette = 1.0f - pow(distance / maxDistance, 8);

```
// Apply vignette to sepia values newBlue,newGreen and newRed
```

```
newBlue = std::max(0, std::min(255, (int)(newBlue * vignette)));
newGreen = std::max(0, std::min(255, (int)(newGreen * vignette)));
newRed = std::max(0, std::min(255, (int)(newRed * vignette)));
```

- **6. A Time your first implementation using time blur function on the included image. Include the timing information in your report.**
  Time per image for blur5x5_1: 0.0891 seconds

- **6.B Include your timing information in your report and explain what you changed to make the filter faster.**
  Time per image blur5x5_2: 0.0188 seconds

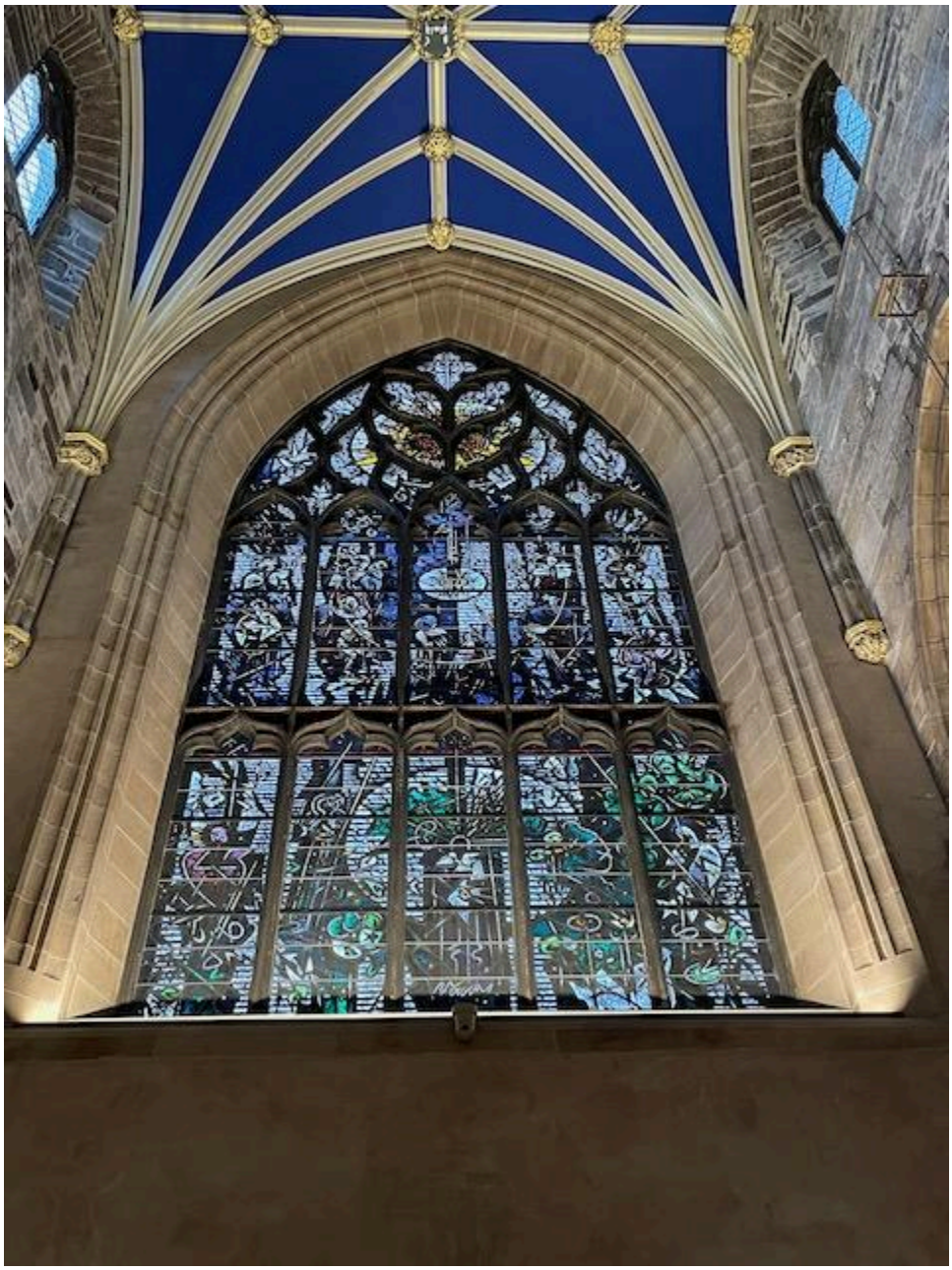
  Separable Filter:
  Uses two 1D convolutions instead of one 2D, reducing operations from 25 to 10 per pixel

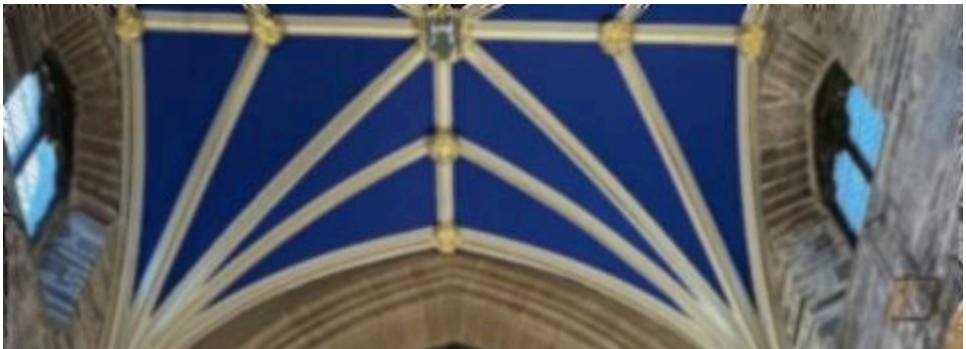
  Direct Memory Access:
  Uses ptr() instead of at(), faster memory access with pointer

  Channel Processing:
  Processes channels directly using offsets, avoids Vec3b conversions

  Single Kernel:
  Same 1D kernel [1 2 4 2 1] used for both vertical and horizontal passes

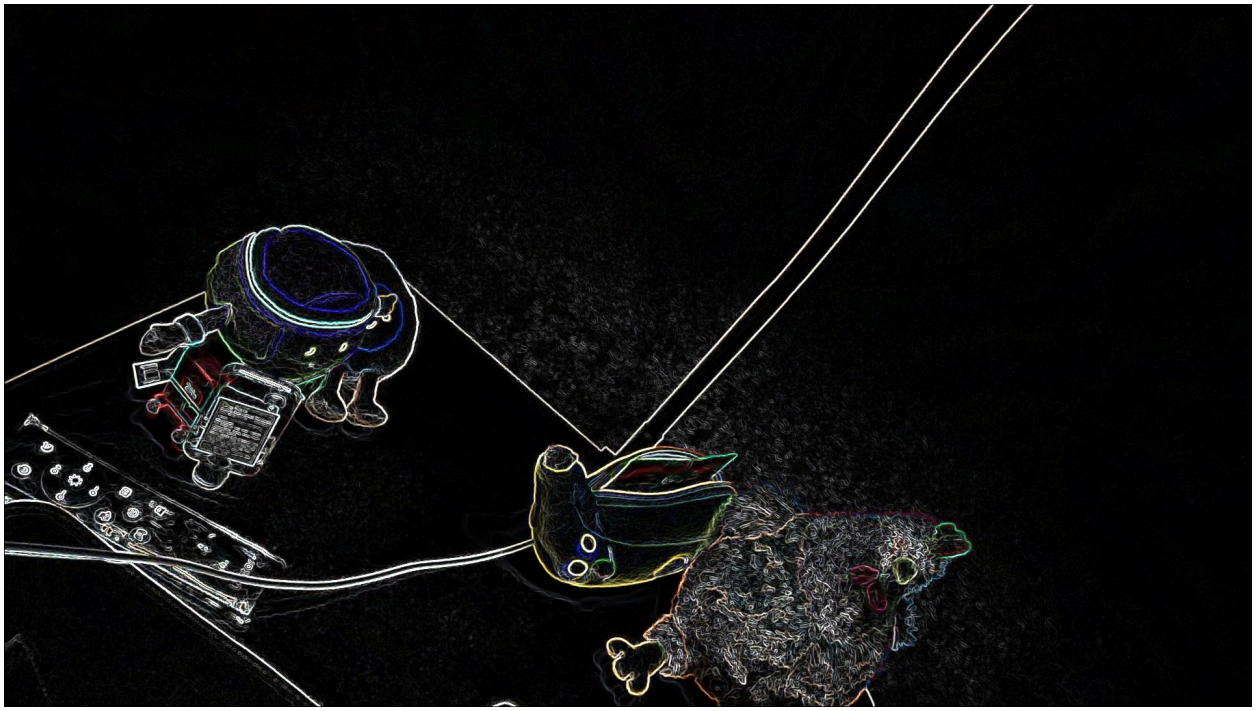
- **Required Image 4: show the original and the blurred image in your report along with the timing information.**





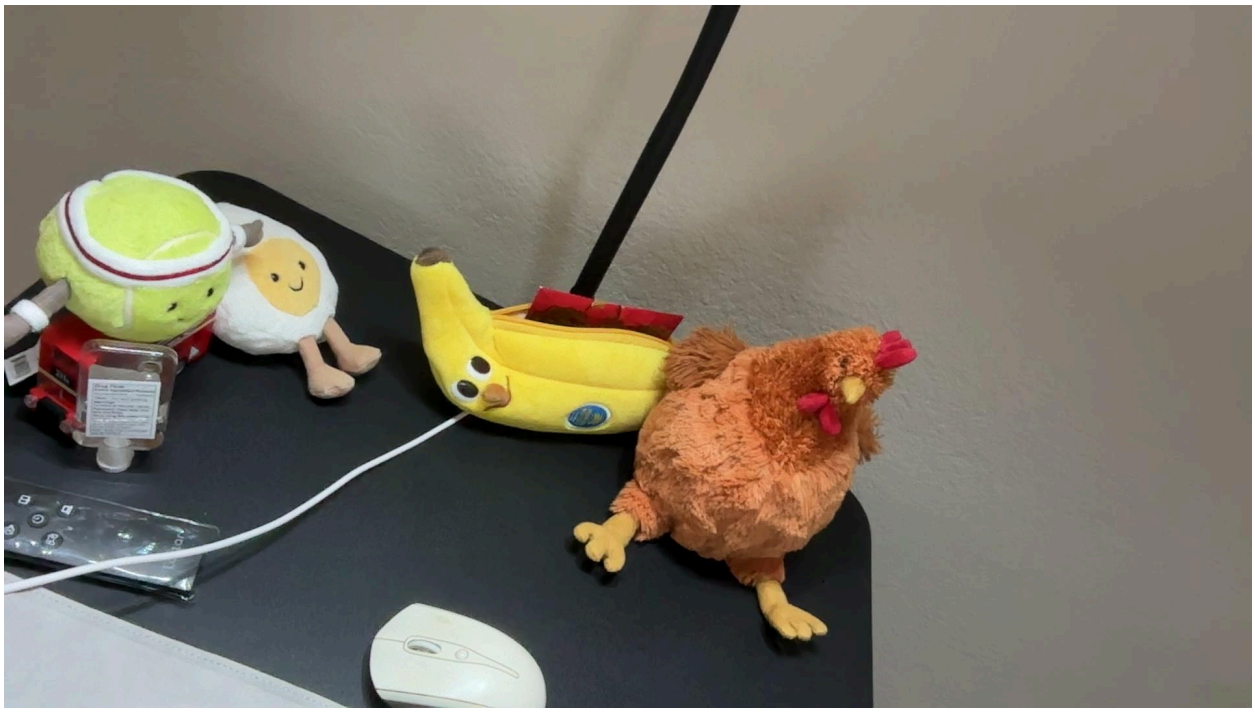
- **Update your vidDisplay.cpp so that if the user types 'b' it uses this function to generate a blurred version of the video stream (in color).**

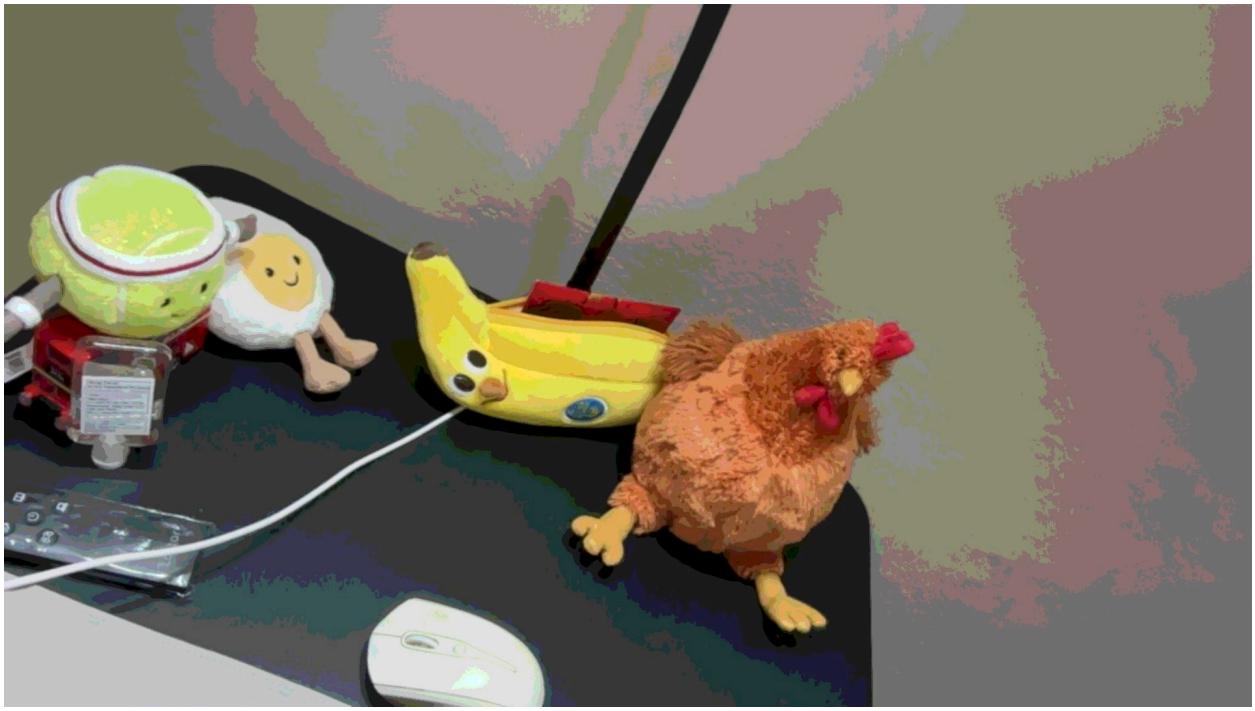
- **8. Required Image 4: show the original and the gradient magnitude image in your report.**



That’s cool!

- **9. Required Image 5: show the original and the blurred/quantized image in your report.**

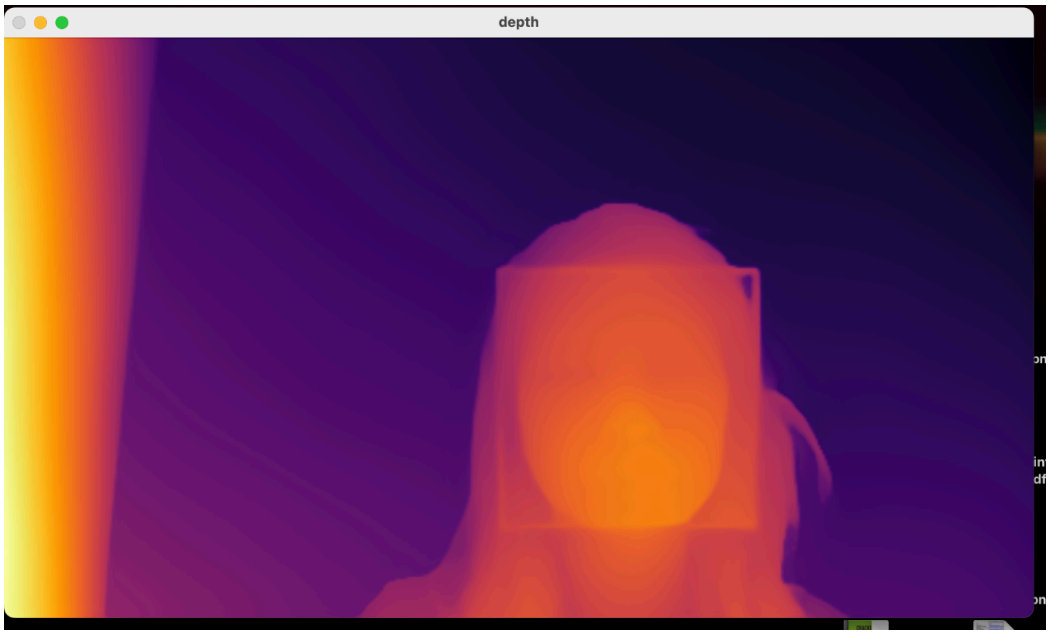




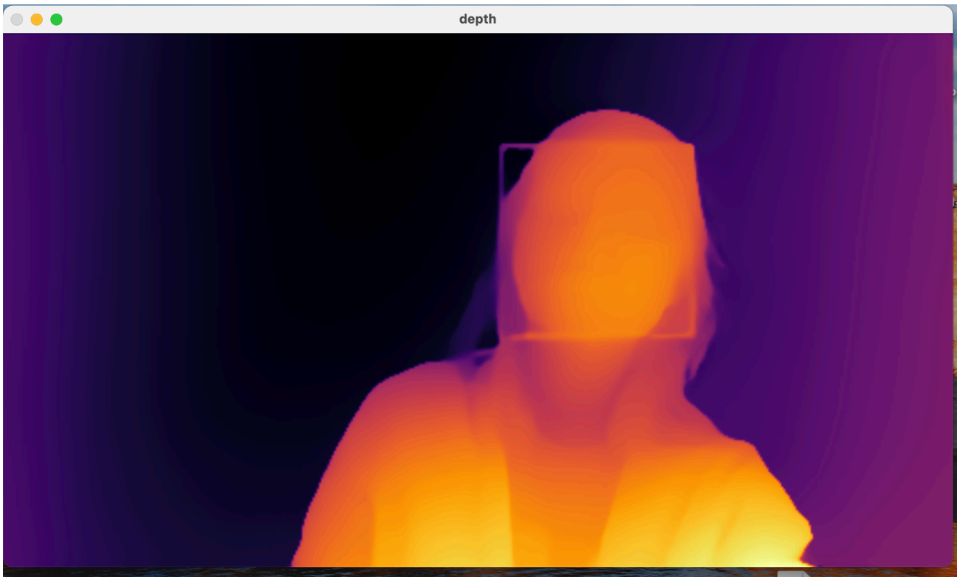
- **10. Required Image 6: show a face being detected in your video stream.**

- **11. Required Images 7-8: show a depth image from your video stream and an image of your filter using depth values.**

  Identify the distance to the face.



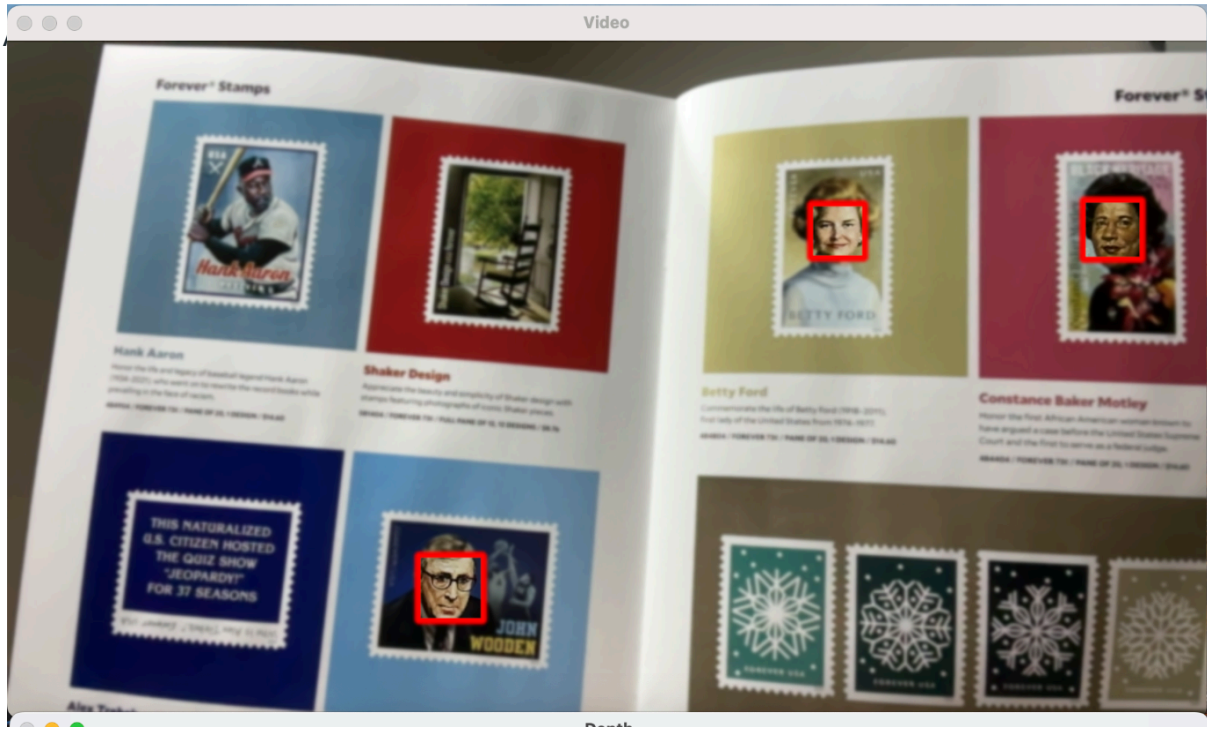
Use blur5x5_2 function (5x5 Gaussian blur using separable filters)

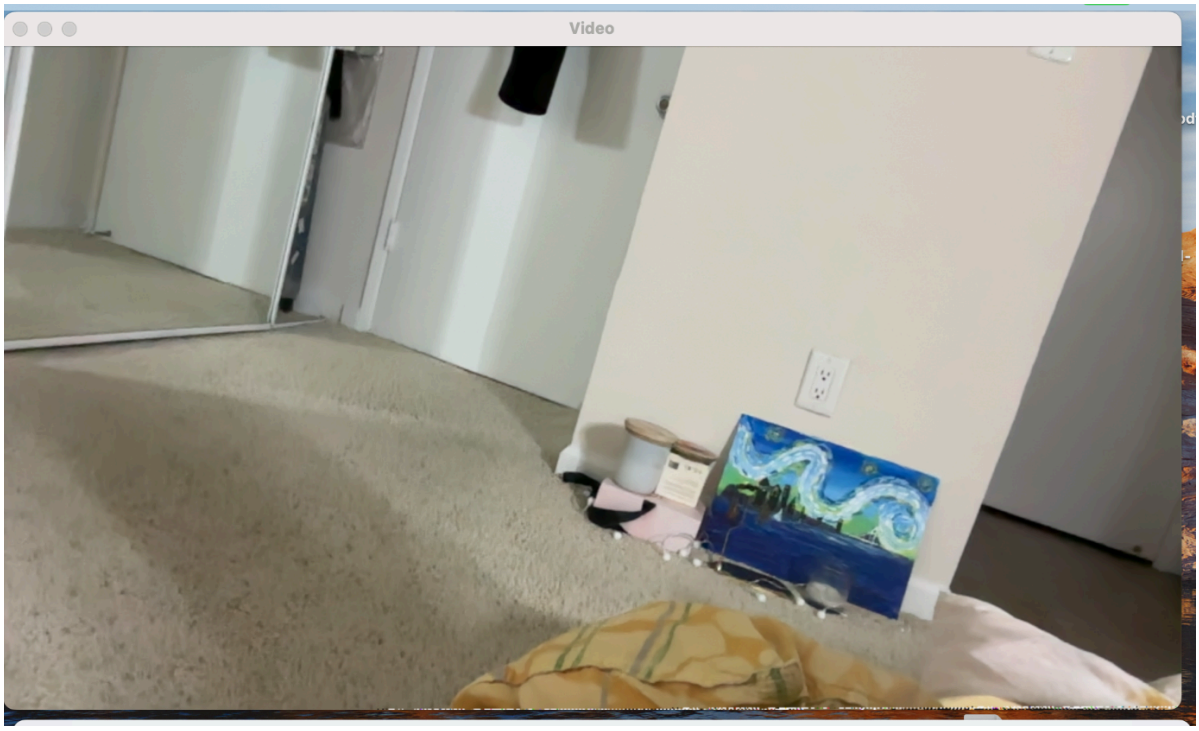


- **12. Required Images 9-11: show the original and the modified images in your report.**

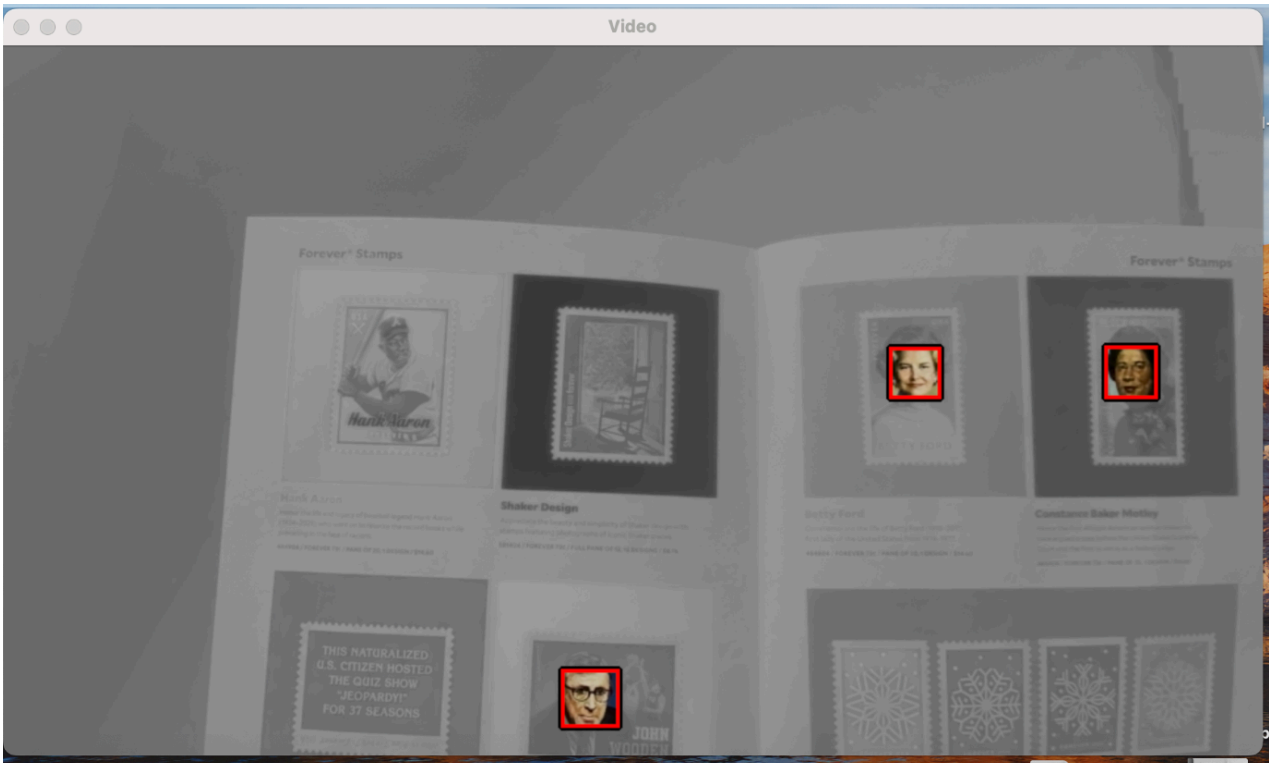
  Extension: in the da2-video.cpp file, I implement effects for video and enable the user to save the modified images by pressing key ‘s’

  Blur the image outside of found faces.





Make the face colorful, while the rest of the image is greyscale.



Acknowledge

https://northeastern.instructure.com/courses/206145/files/31639215?wrap=1
https://docs.opencv.org/4.5.1/d9/df8/tutorial_root.html
https://northeastern.instructure.com/courses/206145/files/31639244?wrap=1
https://northeastern.instructure.com/courses/206145/files/31639297?wrap=1
https://northeastern.instructure.com/courses/206145/files/32259819?wrap=1