Face Mask Detection

Created by Garima Maheshwari and Hailey Schauman

Face Mask Detection

Software that uses object detection to determine if a person is wearing a mask.

Logic Flow

1. Read in exemplar and search image
2. Perform edge detection
3. Create transformation space for exemplar image
4. Iterate through the search image
5. At each iteration, go into transformation space and perform divide and conquer
6. Find highest count
7. If passed a threshold, there is a match

Algorithms

Edge Detection

To convert the exemplar and search image into edge detected images.

Using OpenCV:

1. Convert to a gray-scale image
2. Perform a Gaussian blur
3. Perform edge detection using Canny function

Edge Detection

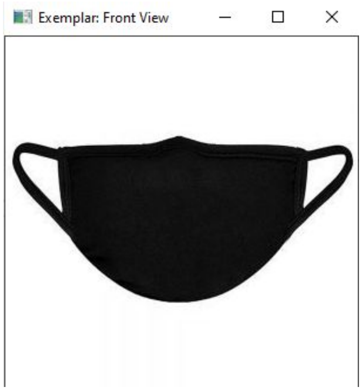
Calculated values

- Minimum and maximum threshold for the Canny function
    - Compute gray average (Wong)
    - Minimum threshold: average - 60
    - Maximum threshold: average + 30

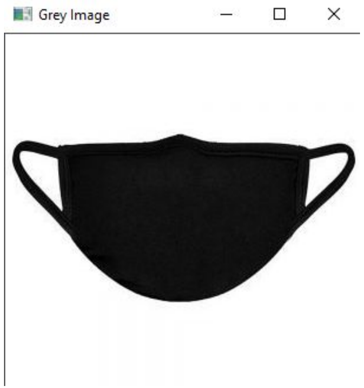
Hard-coded values

- Gaussian Blur function
    - Kernel size of (3, 3) for all images
    - xSigma: 2
    - ySigma: 2

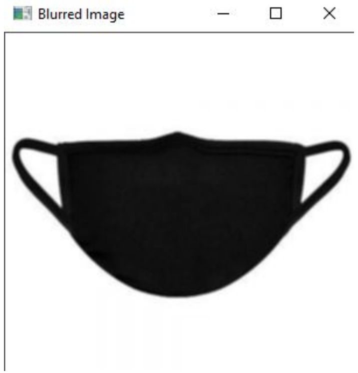
Exemplar Image



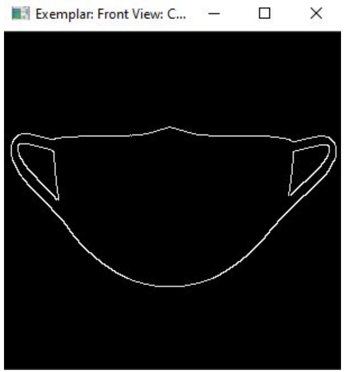
Original



Gray-scale

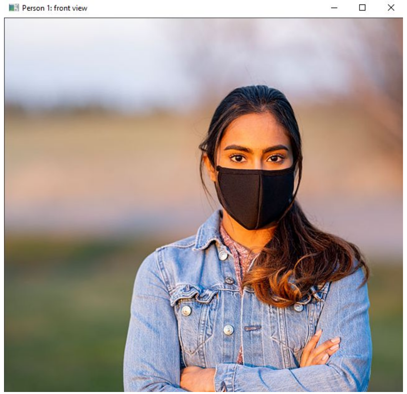


Gaussian Blur

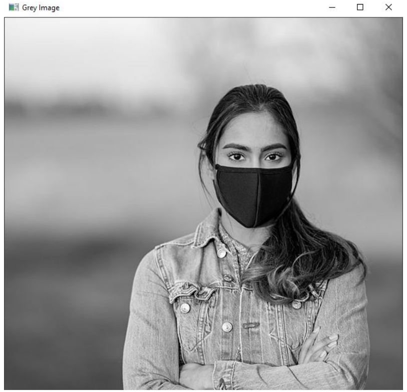


Edge-detection

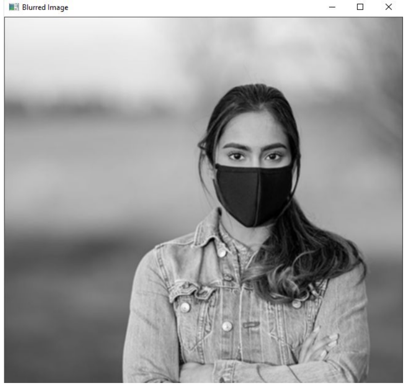
Search Image



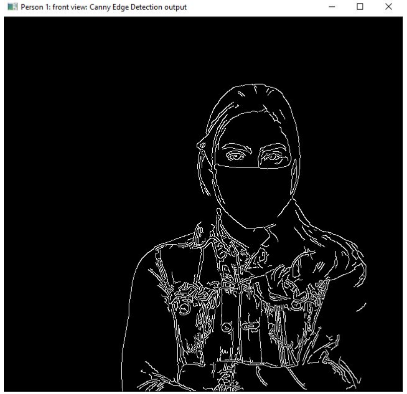
Original



Gray-scale



Gaussian Blur



Edge-detection

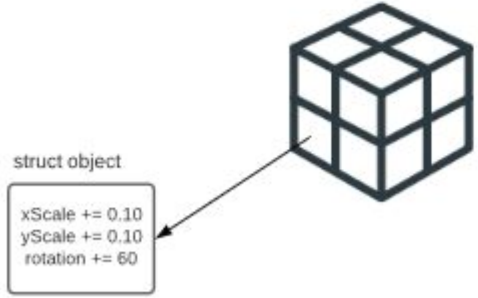
Object Recognition

To iterate over the search image with the exemplar to see if there is a match.

1. Create transformation space for exemplar
2. Detect boundaries on search image
3. Iterate through search image from boundaries with exemplar image
4. Divide and conquer
5. If possible, find a match

Transformation Space

- Determine x, y, and z size for vector based off exemplar size
- Create a struct object that holds
  - xScale
  - yScale
  - rotation
- Increment values at each iteration
- Store into a 3D vector



struct object

xScale += 0.10
yScale += 0.10
rotation += 60

Detect Boundaries with Search Image

- Detected the edge boundaries of a search image
- Used these boundaries as starting points for image iteration
- Decreased time complexity

Cropped Search Image

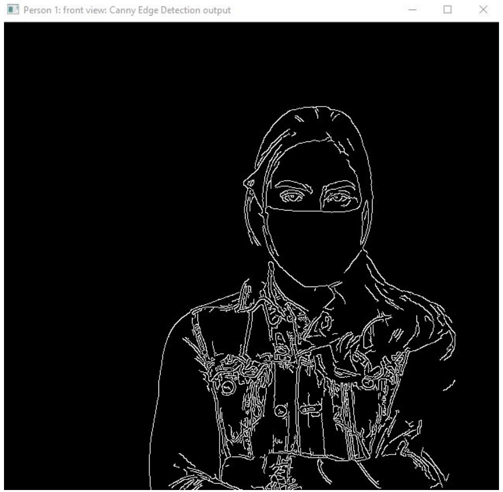


Figure 1: Before cropping

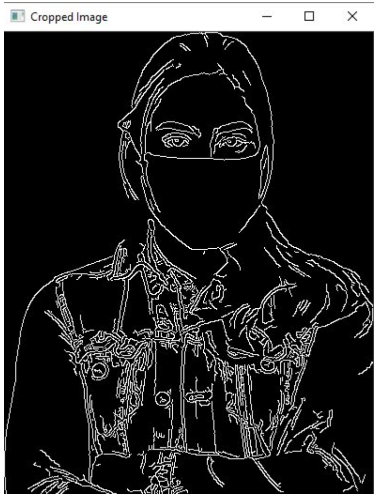


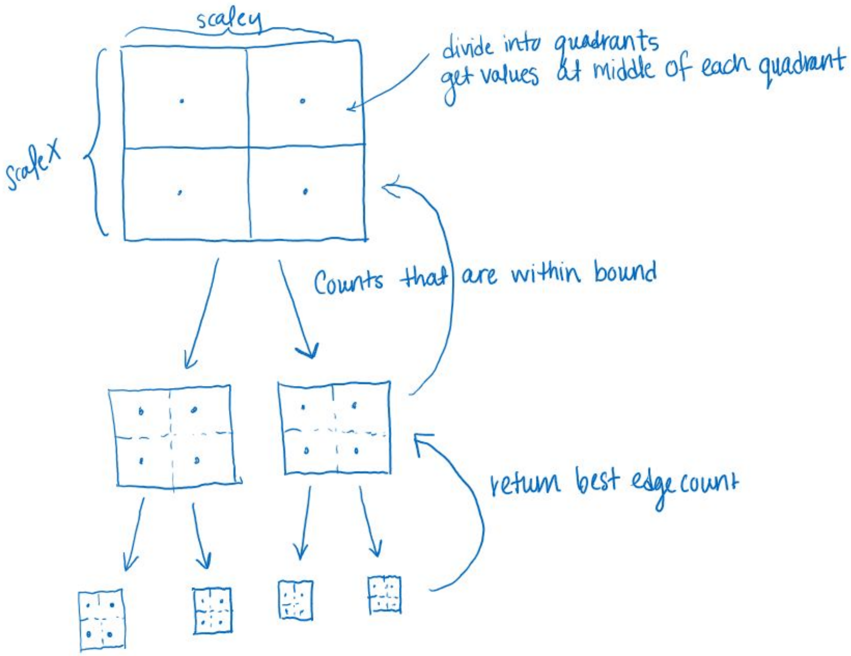
Figure 2: After cropping

Divide and Conquer

- Split scaleX and scaleY transformation space into 4 quadrants
- Find counts at middle of each quadrant
- Split into smaller quadrant if edge count is within calculated bound
- Keep splitting each quadrant until best count number within threshold is found

Note: Since we were working with scaling transformations our bound had to take scaling into account when comparing to exemplar image edge counts.

Divide and Conquer


scaley
scalex
divide into quadrants
get values at middle of each quadrant
Counts that are within bound
return best edge count

Current Testing

Exemplars

- Cotton Masks
- Front-view perspective of masks

Search Images

- Positive testing: front-view of one person with mask per image
- Different colored masks
- Negative testing: images with no masks, but many edges

Results

First Test

Used an exemplar image to detect itself using divide and conquer.

Figure 1: Exemplar used with 1102 edges

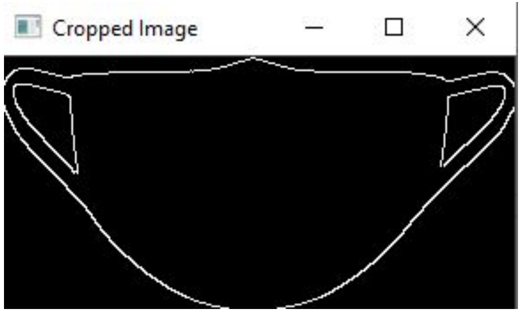
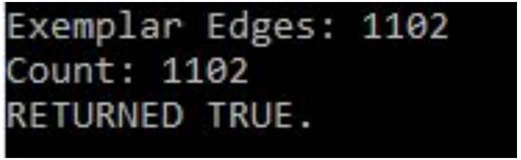


Figure 2: Exemplar successfully finding the search image



```
Exemplar Edges: 1102
Count: 1102
RETURNED TRUE.
```

Positive Test

Used an exemplar image to detect itself using divide and conquer.

Figure 1: Exemplar and search image used

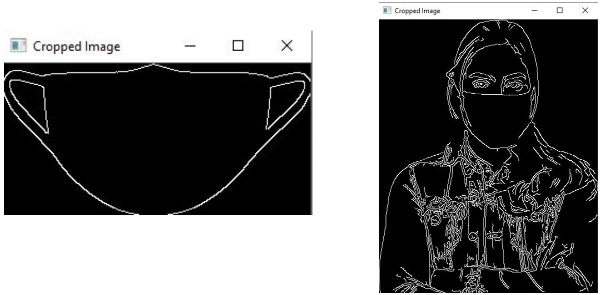
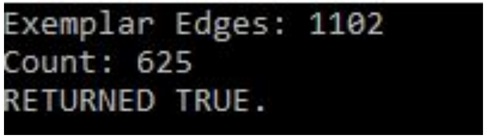


Figure 2: Results



```
Exemplar Edges: 1102
Count: 625
RETURNED TRUE.
```

Negative Test

Used an exemplar image to detect itself using divide and conquer.

Figure 1: Exemplar and search image used

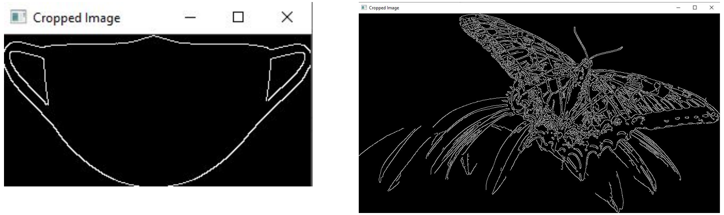
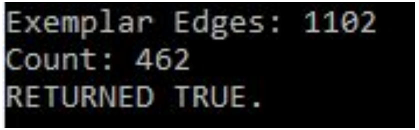


Figure 2: Results



```
Exemplar Edges: 1102
Count: 462
RETURNED TRUE.
```

Constraints

Difficulties with Determining the Bound

- Scale changes led to number of edges to change - difficult to determine in relation to exemplar image
- Difficult to find number of edges in relation to different scaleX and scaleY values
- Hard to determine the bound with ratio - led to inaccurate calculations (had to hard-code temporarily)

Future Implementation

Priorities:

- Allow for more efficiency and faster results: Divide and conquer for all translations
- Show results that are say whether the person is wearing a mask or not
- Develop an improved algorithm for bound

Additional Things:

- Different perspectives of masks (side view)
- Lip detection: Are the person’s lips covered?

Future Testing

Priorities:

- Incorporate side view perspectives of the cotton masks
- Determine what a “good” count is

Additional Things:

- Multiple people with masks
- Various of colors and shapes of masks

References

- Wong, Kerry. “Canny Edge Detection Auto Thresholding.” *KerryWong*, 7 May 2009, www.kerrywong.com/2009/05/07/canny-edge-detection-auto-thresholding/. Accessed 30 Nov. 2020.