



ABSTRACT

Discuss the design, implementation, and effectiveness of a Machine Vision facial classification algorithm


Allison Christensen
10211533

Thomas Heysel
20000838

Declaration of Originality

We certify that Allison Christensen and Thomas Heysel are the sole authors of this project, which was jointly developed through pair programming.

Allison Christensen: 

Thomas Heysel: 

Description of Method and Rational

The objective of this project is to develop a machine vision algorithm that can differentiate between a set of images of a real face, and a set of images of a picture of a face. Each image set, or triplet, consists of 3 images: a frontal view, a left profile view, and a right profile view. Our approach to determining if a triplet is real or fake is as follows:

1. Detect the faces using a Haar Cascade Classifier.
2. Create Region of Interest (ROI) images including only the detected face.
3. Detect keypoints and descriptors between the central and left images and the central and right images using a SIFT detection algorithm.
4. Detect matches within the two pairs of images using a BF Matcher.
5. Filter matches using a Lowe's Ratio of 0.8 threshold.
6. Classify the triplet as either real or fake.

The classification decision is based on a 2-factor confidence level. The first factor comes from the Haar Cascade facial detection algorithm, the second comes from the Lowe's ratio of keypoint matches.

Let's first look at the Haar confidence measure. When a real face triplet is passed into the Haar Cascade, the left and right faces require a profile cascade for detection, whereas the central face requires a central cascade. After initial testing, we realized that for the fake left and right faces, the profile cascade is ineffective at detection. This is because it is not actually a profile; it is a central face that has been tilted. We used this discovery to our benefit. As such, our algorithm first tries the left and right images on our profile cascade detector. If this results in a detection, it raises our confidence that the image is real. If there are no detections with the profile cascade, we then try the frontal cascade detector. If the frontal cascade results in a detection, it increases our confidence in the image being fake.

The second confidence measure is computed from the number of keypoint descriptor matches remaining after Lowe's ratio filtering. We predicted that the fake images would have far more matches as fake triplets are composed of the exact same image. Conversely, real images would have far fewer matches as the profile keypoints and frontal keypoints would have very little overlap. After some initial testing we confirmed that this was the case. When comparing either the left or right image to the central we found fake images had on average between 130 – 200 matches, whereas real images had on average anywhere between 5 – 30 matches. We set a threshold at 80 and stated that any triplet with more matches than the threshold raised our confidence in it being fake. Any triplet with fewer matches

than the threshold raised our confidence in the images being real. While this worked well for the given dataset, for the one we constructed there were different ranges of matches for real and fake images. We determined that a much better indicator is the ratio of number of matches to number of keypoints. A perfectly duplicated image will have a ratio of 1 – where every keypoint has an associated match. In practice, we found our fake images to have ratios ranging anywhere from 0.2 – 0.5. Real images have a ratio in the range of 0.001 to 0.09. We set a threshold at 0.2 and stated that any triplet with a ratio greater than the threshold raised our confidence in it being fake. Any triplet with a ratio less than the threshold raised our confidence in the images being real.

When both checks are finished whichever confidence level is higher, real or fake, becomes our classification. If the confidence levels are the same, we classify it as fake as we assumed the application of this algorithm would be biometric identification and false negatives have a lower threat to security than false positives.

Description of Implementation

Our implementation makes use of OpenCV's cascade classifier objects and Haar cascade classifier xml files for both the frontal and profile faces. We also use OpenCV's SIFT and BF_Matcher objects for keypoint detection and feature matching.

One issue we ran into was the Haar cascade detecting small inanimate background objects as faces which resulted in incorrect classifications. We were able to remedy this issue by specifying the minSize argument in OpenCV's facial detection function detectMultiScale. By setting a restriction on the minimum size faces could be, we effectively solved the issue of misclassification in the given dataset and got the unintended benefit of faster runtimes. This performance increase was due to the fact that the Haar Cascade no longer had to convolve the images at small mask sizes.

Another issue we ran into was with feature detection in the test data set we created. Due to a combination of factors such as bad lighting, poor image resolution, and poor printer quality, we found that our fake faces had few distinguishable features. One factor that helped reduce the impact of this issue was equalization before feature detection. Once the face has been detected and extracted, we convert it from RGB to YUV, a format that includes the color as well as luminosity. This allowed us to convert the YUV image into a histogram, equalize it, and then convert it back to RGB color. This equalization increased the contrast in the facial images, making our SIFT object able to detect more features, which resulted in better classifications.

Description of Tests

To test the effectiveness of the algorithm we first used the given test dataset and then supplemented it with our own test images. The test data set our team built consisted of 81 separate images, 27 triplets. Unfortunately, our test set is uneven, with 12 fake and 15 real triplets. We were aiming to have a fake triplet for every real one but we ran out of printer ink.

We were interested in testing the effectiveness, not only on different people, but also with different facial expressions. The dataset we built consists of neutral, happy, sad, angry, and goofy faces. The below Confusion Matrix as well as the following precision and recall equations were used to analyze the test results:

Table 1 Confusion Matrix Structure

Number of True Negative	Number of False Positive
Number of False Negatives	Number of True Positives

$$Precision = \frac{N_{TP}}{N_{TP} + N_{FP}}$$

$$Recall = \frac{N_{TP}}{N_{TP} + N_{FN}}$$

These results are documented in Table 2, as seen below:

Table 2 Test Results on Different Datasets.

Dataset	Precision	Recall	Confusion Matrix		Number of Unclassified Triplets *	Runtime (ms)
Provided Data	1.00	1	8	0	5	3.042
All Supplemental Data (all 5 facial expressions)	0.94	1	0	11	0	2.885
			11	1		
Neutral Supplemental Data	1.00	1	2	0	0	0.498
Happy Supplemental Data	1.00	1	0	3	0	0.499
			2	0		
Sad Supplemental Data	1.00	1	2	0	0	0.556
Angry Supplemental Data	0.75	1	0	3	0	0.646
			2	1		
Goofy Supplemental Data	1.00	1	3	0	0	0.669
			0	3		

* Unclassified triplets indicate events where in 2 or more of the 3 images the Haar Cascade is unable to detect a face and the algorithm therefore does not have sufficient data to make a classification.

Effectiveness of Solution and Limitations

As reported in the previous section, the recall value maintains a consistent score of 1.0 across all datasets. This indicates that the algorithm is very effective at classifying fake images. Precision scores

across all datasets are relatively high which indicates that the algorithm is effective at classifying real images. The angry supplemental dataset is the only set that did not achieve a perfect precision score. This suggests that the algorithm is vulnerable to incorrectly classifying a fake image as real, meaning if this algorithm was used for security through biometric identification, it could be susceptible to fake image attacks.

Overall, the system is successful, and this behavior is expected to be consistent on images that meet the following criteria:

- The facial features in the image are fully in view (eyes, nose, forehead, mouth, chin)
- The image subject's face is in either a central or profile position
- The image brightness does not obscure normally high contrasting regions of the face

Limitations

The main limitation observed in the system is its inability to classify every input triplet, as seen in the *Number of Unclassified Triplets* column in Table 2. A triplet is deemed unclassifiable if the algorithm cannot detect a face in 2 or more of the images, as we need at least one profile and the central view to make a feature map. A recurring detection issue that we encountered was in images where the subject was wearing glasses and light is reflected in the lenses. Facial detection with Haar Cascades relies on the property that the eyes are darker than the bridge of the nose or the forehead. However, when light is reflected in the subject's glasses, their eyes appear lighter than expected and this property no longer holds; the detector is therefore unable to detect a face based on this feature. While histogram equalization was able to mitigate some instances of image brightness washing out regions of expected contrast, it could not resolve detection issues in this specific scenario.



Figure 1 Unclassifiable image example

Another limitation in the system is that it requires high quality images for accurate fake face classification. As discussed previously in the Implementation section, the test images we created were low quality, which resulted in low numbers of detectable features and therefore resulted in more False Positives.

Finally, the system is limited to classifying images with central or profile faces, as these are the only positions for which cascade classifier .xml files are loaded. The system has not, for example, been tested on images of upside-down faces, and is not expected to achieve the same performance with these deviations.

Improvements

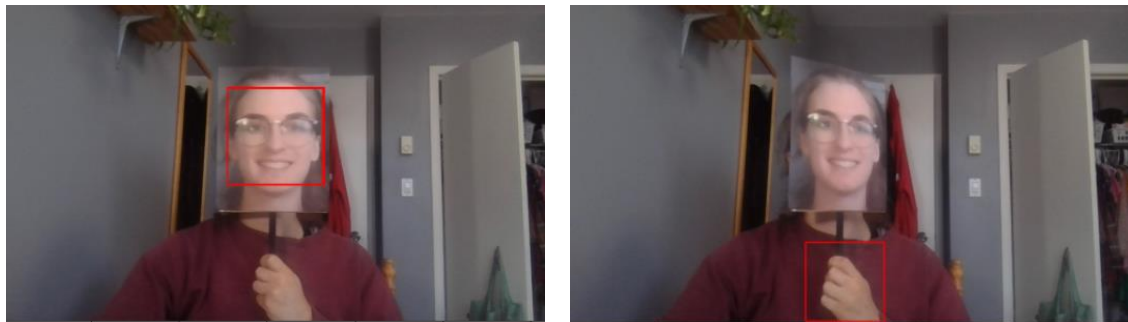
Given more time to work on the project, the current system could be improved by increasing the test set to include participants of varying ages, ethnicities, and facial features (facial hair, glasses, makeup, piercings etc.). Expanding the test set would help to both improve the robustness of the system and to avoid excessive bias in the system towards a single demographic. Once we have a more complete data set, we would also want to incorporate varying levels of translation, rotation and scaling to the training images.

Outside of this course, i.e. with no library restrictions, we would be interested in investigating the effectiveness of building a Convolutional Neural Network (CNN). It has been accepted in the machine learning community that computers achieve far better classification results when they are allowed to learn themselves, as opposed to rule-based classification such as the algorithm we built. CNN's are especially effective in machine vision applications as they greatly reduce the number of parameters compared to a standard fully connected neural net. High quality images such as those we are using to test the system can have several thousand parameters (ie pixels).

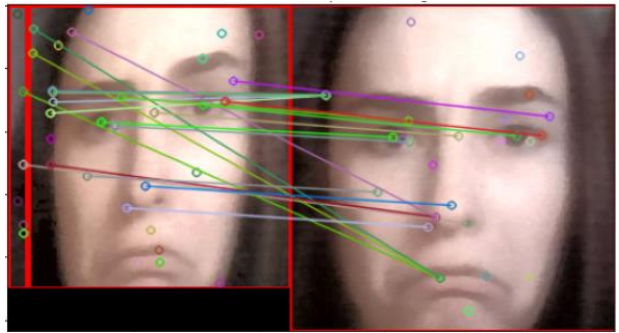
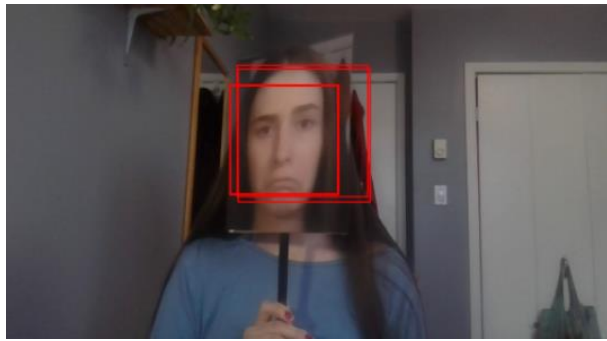
Resolving Known Issues

Finally, if given more time we would want to overcome some of our system's limitations by resolving our known issues. Note that because of the 2-factor confidence ratio, even when these issues occur, most of the images are still classified correctly. Nevertheless, these are cases of unexpected behavior that we have identified but have not had the time to resolve.

Moving forward, we would work on improving facial detection. We have observed some instances of incorrectly detected faces, as seen below:



Another issue we would want to resolve is the detection of multiple faces. In the example below we can see multiple bounding boxes around the detected face. As a result, the smaller bounding boxes are visible in the larger facial ROI. This confuses the feature matching, as the red from the bounding box is matched with the red of the face's lips.



Finally, we would want to correct an issue related to the tightness of the bounding box size relative to the subject's face. In the example below, we see that part of the door behind the face is included in the bounding box. This results in the edge of the door being labelled with feature descriptors that are being incorrectly matched to the collar of our housemate's shirt.

