CSE 455: Computer Vision
Final Project Report: MemeDub

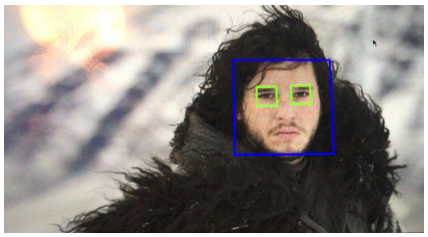
Zhaoyuan (Matt) Xu xzy1998@uw.edu
Chenyang (Michael) Fang chenyf@uw.edu
Andrew Wei nowei@uw.edu

Abstract

We put UW hats on faces for images and UW glasses on faces for videos in real time like a “Snapchat” filter. We did this using a mix of facial feature recognition and math. We originally wanted to only put hats on images, but we felt like we should extend the project so that it projects images in real time based on the location of the facial features we track.

Inspiration

We, as UW students, subscribe to memes made specifically for UW. A large number of them can be found on the UW Teens for Boundless Memes group on Facebook. There is a subset of memes in the group that involve UW hats to signify membership within the UW community. In honor of this specific subset of memes, we wanted to create something that other students at UW could use to reduce the amount of time it takes to make these memes.



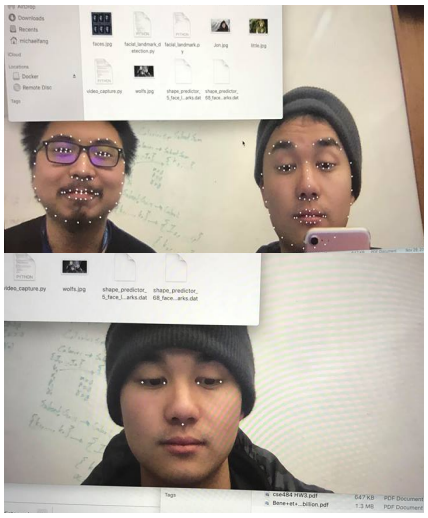
Implementation

Our project is divided into two big challenges:

1. Figure out where the face is from an image.
2. Draw a proper hat for that face base on some parameters.

For the first challenge. We initially tried to use OpenPose to get an idea of what type of framework we wanted to use to accomplish this project. After installing OpenPose and trying out their demo, we got an idea of what features we wanted to keep track of in order to accomplish our task. We eventually moved away from OpenPose because the framework was a more complex than we needed, but we figured out that we wanted to identify 3 points on the face, i.e. the outer corners of the eyes and the base of the nose.

To do this, we used the a pre-trained model that was trained on the iBUG300-W dataset. We decided to use a pre-trained model because we didn’t have the time to train it ourselves. We did not

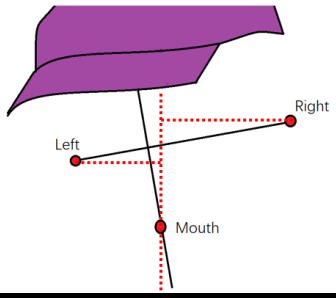


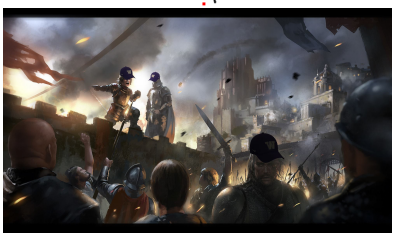
have the time to train our model due to it being finals week and members within the group having finals early in the week of finals week.

For the second challenge, our goal is that given three points about the face: left and right corners of the eyes and the nose as shown in the picture below, we can then compute the the parameters for the hat and draw them on each face.

We ended up working on the first and second challenges simultaneously, so we had to manually enter the location of the face to test out how to put a hat on the image. Later, we wrote a bridge script that can redirect the face location data from our python program into the C program that draws the hat.

Our hat drawing workflow is: We firstly use the horizontal relative location to determine where the hat should be pointing towards and which hat should be put on. Then we calculate the size of the face and figure out the proper size of a hat using some expansion parameter. As for resizing we used the bilinear resizing method we implemented in the class. Using the tilting angle calculated during the face size calculation, we can figure out the the rotation angle of the hat and rotate it using our own rotation method. After that, we analyze the pixel value at the three points to gain the brightness suitable for the hat by temporarily converting the image from RGB to HSV and extracting the Value using the image processing library we implemented during this quarter. At this point, the hat is processed and we can start to draw them on the face. We use the three locations and some tuned parameters to figure out the center of the hat, and we draw the hat on it.



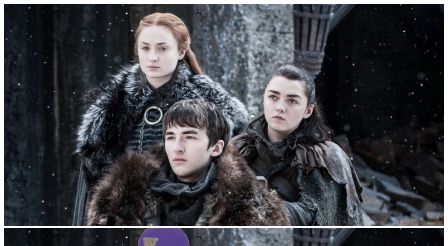


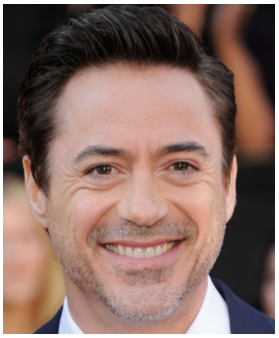
Below are the results of manually entering the face locations. As shown below, the one after our modification (Bottom one) looks more natural than the one that simply gets pasted (Top one).

After tuning our parameters to be more accurate, we started to do the combinations with our face detector and polished it more. The results after these adjustments are shown in the results section.

Results

The hat putting on images is pretty good for some pictures, but it also has glitches.



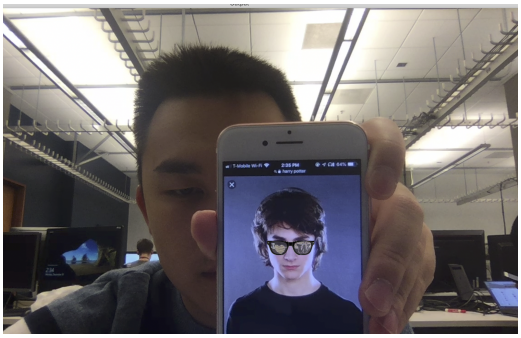



And as shown above, there are still some glitches, and Bran's face was not detected on the bottom left set of images, which is something we need to work on in the future. Also, our face detector cannot detect very small faces or very dark faces (like the images below, which we also need to work on in the future).


We think the dark face is failing because it makes the features needed to be recognized as a face too similar to the surrounding, making it hard to be distinguished, or it could be the case that the dataset we used was biased, and it was trained using mostly pictures of lighter-skinned people. As for the small faces, we think it is because the features are too small to be recognized together to form a face that can be recognized by our face detector or their heads were tilted in a way that makes it unrecognizable to the model.

Extra Feature: Video Capturing

We eventually realized that we were actually just making snapchat filters when running our code for each frame captured by the camera. It presents more challenges. Although it is an extra feature to our project, we spent lots of time and energy on it



Challenges

1. First and foremost, we have the same issue of detecting the faces accurately since we are using the model provided by dlib
2. To challenge ourselves, we chose to try putting glasses on people instead of hats. The results are as below. However, different from hat, we could not find pictures of glasses from various angles. Therefore, our model works when human face is facing the camera. We have come up with multiple strategies to try to solve the problem. Since people in the video may face the camera

from a large range of angles. We agreed that a 3D-Constructed model of glasses would be optimal. However, we don't have the resources or time to apply that idea.

3. The third challenge was resizing and rotating the image, which is the most challenging one for us. To overlay a pair of glasses over the frame of a person, we had to figure out how to resize our glasses and how much we had to tilt it (since a person might tilt his or her head). Thanks to the landmark detector, we could obtain the information we need. However, we are unable to finish a function that could successfully tilt the glasses(you cannot just loop from the first pixel and overlay that on the frame).

Accomplishments

1. We have successfully used the camera feature provided by OpenCV and applied the landmark detector to capture features on the face
2. We have successfully retrieved the information that we needed to resize and tilt the glasses
3. We have successfully resized our glasses according to the size of the faces
4. We have successfully overlayed our glasses on each frame and made the background of the glasses picture transparent when we try to place the glasses on human face

Future Work

Firstly, our current face finding cannot detect faces that leans towards one side too much or the faces that are very small in the image. Therefore, we need a better face detecting tools, and we are considering digging deeper into OpenPose to utilize their library. Also, our algorithm for hat putting is still rather primitive, and it would be better to research more to generate a more suitable computational model that can also be computed relatively fast because our final goal is to make a realistic hat-wearing experience in real-time using the webcam. We may also train our own model in the future to see how it stacks up to the current one and see if we can improve the range of faces it can detect (rotated and darker faces).

Conclusion

In conclusion, we can now put UW hats on human faces on an image relatively quickly, but the accuracy still needs more tuning. Our success of adding accessories in real-time using our webcam gave us insight on how we could potentially port our hat-adding program as a real-time Snapchat filter.

References

- Facial mapping (landmarks) with Dlib + python by Italo José
- C image processor by UW Computer Vision Course of Autumn 2018
- UW Teens for Boundless Memes
- C. Sagonas, E. Antonakos, G, Tzimiropoulos, S. Zafeiriou, M. Pantic. 300 faces In-the-wild challenge: Database and results. Image and Vision Computing (IMAVIS), Special Issue on Facial Landmark Localisation "In-The-Wild". 2016.