SNAPCHAT FACIAL FILTERS PROJECT
AUGMENTED REALITY CS7434

EDMOND O' FLYNN
12304742

Table of Contents

INTRODUCTION

ABSTRACT

Snapchat has become a staple in the modern world of social media. In recent months, Snapchat introduced an augmented reality camera lens filter system that has enraptured users with its quirky and humorous effects. This project aims to explore the world of camera calibration to find known facial landmarks, to calibrate the orientation of the face, to generate a pose, and to apply various effects to the user's facial structure spanning flat portable network graphics distortion and applications. In addition to this, the project also aims to tackle the filter known as a face swap using the Orthogonal Procrutes Problem as a method to handle and mitigate orthogonal calibration issues for swapping facial features with another image of a differing projection.

To understand how this augmenting of reality works, the process of generating a known calibration in 3d from a 2d image representation of the world must be traversed. Using various methods of detecting the locations and classifications of features, a temporal perspective axis can be generated and used to augment and distort these effects to the desired location on the user's face. This project aims to explore the area of finding calibrations for the given projection, and apply educated guesses to locating the appropriate markers and orientations for the given features being sought after.

MOTIVATION

Having been an avid user of Snapchat since 2013, the implementation of facial filters interesting from the point of view of pose estimation, calibration, and the application of various lens and distortion effects to the user's face has always been an interesting topic. The motivation for this project has been to understand how this system of calibration works in its most basic forms and to create something interesting that depicts these features using a simple system where an individual can easily add new filters. The overall goal is to create a basic system that can be expanded upon as necessary by others in the future, and to gain a deeper understanding for myself in how such filters work and align themselves with the face's pose.

RELEVANCE

As this feature of Snapchat camera filters has not been made open source through an API officially, it has always left a yearning to create new filters using facial landmarks as anchor points and estimating the face's pose using linear algebra. While this implementation is not as accurate or as efficient as Snapchat's, it provides a solid foundation for implementing simple filters such as glasses and moustaches on any detected faces within the webcam frame, as well as implementing the face-swap feature across faces.

This is relevant to this field as is it as a direct application of Augmented Reality. The method used for super-imposing objects relies greatly on the area of calibration with known facial landmarks, resulting in the augmentation of reality through a camera lens for any faces detected in the camera frame.

THEORETICAL BACKGROUND

CAMERA CALIBRATION

OpenCV uses a projection model known as the pinhole camera (OpenCV, 2013). Using this model, there are several unknowns that are required to convert image points on a plane into object points in real space.

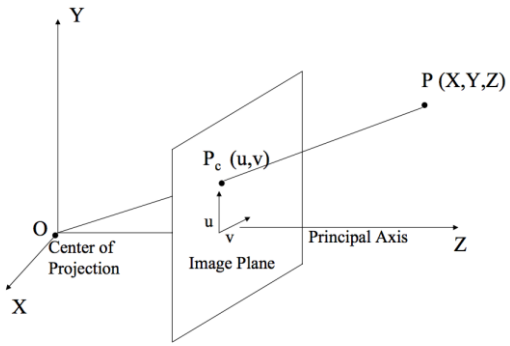

FIGURE 1 PINHOLE CAMERA MODEL (JOSHI, 2015)

The figure above uses a centre of projection, O, and to simplify matters, the principle axis is made parallel to the Z-axis. The image plane is f units away from O, where f is the focal length. The camera plane coordinate pc is at position (u, v, w) where w is constant. Using similar triangles to calculate (u, v) given (X, Y, Z) and f, we come up with the following in homogenous coordinates:

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

Since the Image Plane may not coincide with where the z-axis intersects the image plane, Pc needs to be translated to the desired origin by (t_u, t_v). The new homogenous coordinates are as follows:

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} f & 0 & t_u \\ 0 & f & t_v \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

Finally, the scale of the pixels must be included. If the point P is expressed in cm, and pc is measured in pixels, we need the values mu and mv representing pixels/cm to be included. The equation now becomes:

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} m_u f & 0 & m_u t_u \\ 0 & m_v f & m_v t_v \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} \alpha_x & 0 & u_0 \\ 0 & \alpha_y & v_0 \\ 0 & 0 & 1 \end{pmatrix} P = KP$$

K can then be given a skew factor if the coordinate axis u and v are not orthogonal to each other. This matrix is referred to as the intrinsic parameter matrix for the camera (OpenCV, 2013):

$$K = \begin{pmatrix} \alpha_x & s & u_0 \\ 0 & \alpha_y & v_0 \\ 0 & 0 & 1 \end{pmatrix}$$

HAAR CASCADES

A face has many features that can be used to uniquely classify individuals. Using a large data store of positive and negative images, a cascade can be trained on a data store and classify Haar-like features according to the training received. A Haar-like feature considers only adjacent spatial rectangles in an area being detected, and then giving a difference of the sum of the pixels in the white area is determined from the black area to give the relative feature (Viola & Jones, 2001).

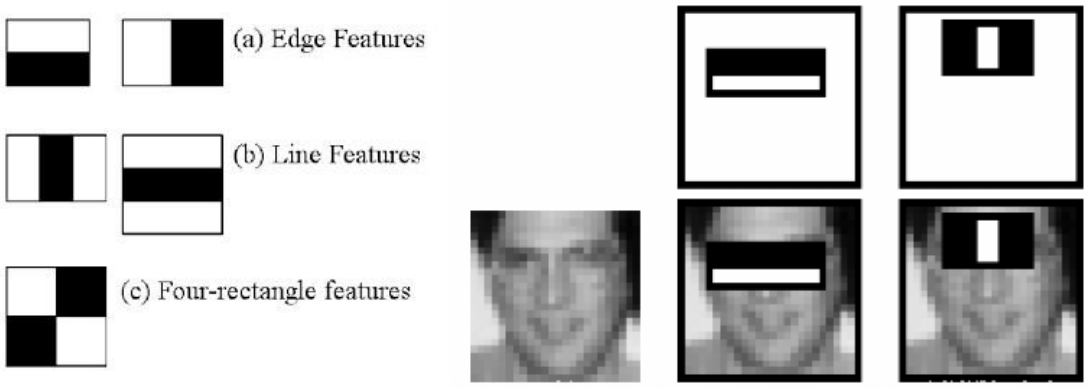

FIGURE 2 HAAR-BASED FEATURES ON A FRONTAL VIEW OF THE HUMAN FACE (OPENCV, 2016)

There are common features across faces that result in various sets of bounding boxes per face – for instance, the area around the eyes is generally darker than the cheek-bone area; therefore, two edge features can be adjacently generated (Papageorgiou, et al., 1998). This detection generates a convolutional kernel that works from pixels within the image itself to give a 2-rectangle feature pixel sum (Viola & Jones, 2001). Sets of features can then be overlaid back onto the face to approximately show the locality of tagged features, i.e. brow, chin, eye, nose, jaw, mouth.

$$area = I(A) - I(B) + I(C) - I(D)$$

The rectangular Haar-like features can be thought of using summed-area tables called integral images. As images can be thought of a matrix of intensity values, integral images can be thought of as a look up table of the same dimensions as the image matrix (Lienhart & Maydt, 1998). A Haar-like feature rectangle consists of 4 points; therefore, a feature requires at least 4 lookups.

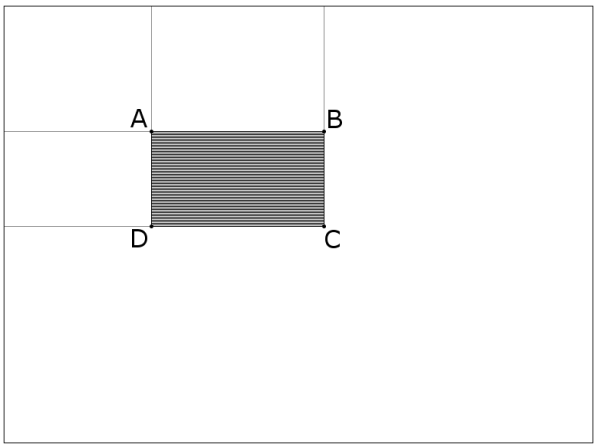

FIGURE 3 VISUALISATION OF THE AREA SUM (MANSKE, 2007)

Facial Landmarks

While Haar cascades generally have a lower computational expense in comparison to other methods, they only give approximate area of features and fail at skewed angles. As more accurate points to accurately get the pose of the face were needed, considering facial landmarks for pin-pointing as the main method of feature detection had to be decided on. DLIB is a modern C++ machine learning framework composing of detection algorithms and tools for solving real world problems (DLIB, n.d.). The framework recently had several new features added, including that of real-time facial landmark detection through one-millisecond facial alignment with an ensemble of regression trees.

The library quickly adds annotations of 68 facial landmarks even at non-frontal, highly skewed angles with accurate detection. DLIB uses the HELEN dataset consisting of 2000 training images of varying lighting, pose, expression, occlusion, and individual differences. (Le, et al., 2012) It is also possible to go beyond the default 68-point generator and create one's own classifier as per Kazemi and Sullivan, where a 194-point model was generated, resulting in a much richer set of points to work with. The main benefit for using facial landmarks lies in the annotation of individual points consistently across skew and varying occlusion. (Kazemi & Sullivan, 2014)

Figure 4 Examples of faces landmark detection at different skew angles. (Earl, 2015)

Anchoring

As DLIB quickly annotates the necessary features, I had to use anchoring points to keep some constants in generating the dimensions of the face and the pose. I decided on using points that are unable to undergo large distortions. As the jaw, mouth and even nose can all distort shape and vary depending on its context, I saw the potential in using the width of the head (points 9 and 16) with the centre point of the nose between the eyes (point 27) as a constant in generating the pose of the face. Face landmarks work from a perfectly centred and symmetrical face, I used this idea to transform and distort from these points to the actual skew and angle of my constant points to get the 3-dimensional pose in a 2-dimensional image.

Anchoring landmarks to a face works by taking the mean shape of facial alignment, which allows for all images to be approximately of the same orientation. This naturally makes sense due to the nature of the logic of more images results in better approximations as per a larger data set. Anchoring also allows for pose estimation of the head, which works to generate an approximation of the amount of available space

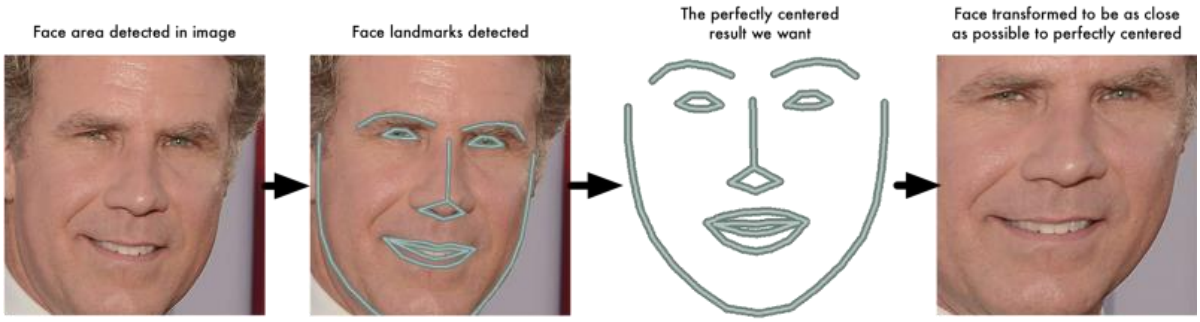

FIGURE 5 DISTORTION OF THE FACE TO IDEAL LANDMARK POSITIONING (MEDIUM, N.D.)

ORTHOGONAL PROCRUSTES PROBLEM

Homography is defined as the relationship between any two planar surfaces shared across images as a bijection of an isomorphic vector space residing in a vector space of points. (Gower & Dijksterhuis, 2004) Images can share the same surface points, but may differ in the perspective matrix. A homography matrix is defined as follows:

$$y_0 = a_{0,0}x_0 + \cdots + a_{(0,n)}x_n$$
$$\vdots$$
$$y_n = a_{n,0}x_0 + \cdots + a_{(n,n)}x_n$$

This idea of finding a collinear mapping between points is important, as it is required for being able to successfully carry out a face swap. Face swaps are an application of homography where a subset of facial landmark points is cropped, masked, and then aligned with the face of the individual to be super-imposed onto. This then leads to an issue known as the Orthogonal Procrustes Problem with the two facial matrices.

In the case of my third proposed filter for face-swapping images, there is a known problem solved in 1964 by Peter Schönemann pertaining to matrix approximation within the scope of linear algebra called the "Orthogonal Procrustes Problem". The problem deals with the issue of being give two matrices A and B, and from them; finding the orthogonal matrix R of mapping A onto B – in this case, the direct application of swapping one face's facial landmark alignment onto another to the closest approximation. (Mount, 2014)

$$R = arg\min_{\Omega} \|\Omega A - B\|_F \text{ subject to } \Omega^T\Omega = I$$

The solution to this problem lies in the minimisation of multiplying matrix B by the transpose of matrix A through reduction to the standard inner product, which results in either the stretching or cropping of an image to the other matrix's dimensions. This means that for this application involving images A and B, the translation, rotation, skew and scale must be determined for one image onto another over all 68 points, i.e.:

$$\sum_{i=0}^{68} \left\| sRp_i^T + T - q_i^T \right\|^2$$

Given that R is an orthogonal matrix for the orthogonal 2x2 matrix of the mapping of A onto B, s is the scaling factor, p is the i^{th} row of image A's landmark, q is the corresponding i^{th} row of image B's

landmark, and T is a 2-vector, this allows for ordinary Procrustes analysis to be determined for the minimisation of one landmark feature onto a corresponding one. (Gower & Dijksterhuis, 2004)

Implementation

Language Choice

Throughout this project, there was the potential to work in many languages; but it was settled on to use Python 3.6. Python is known as the lingua-franca of data science and is a level of abstraction above C/C++. Knowing this, the benefits of rapid-prototyping greatly outweighed the performance drawbacks for expectations within this project.

Haar Cascades Method

Initially, Haar cascades were used as a starting point to explore the area in greater depth. Pretrained cascades were used in OpenCV; detecting the head and nose. As Haar cascades work in a non-skewed fashion with full-frontal views, the position given did not take pose into account. A simple implementation for placing a flat image of a moustache approximately at the located nose in the image was used, taking into consideration only the size of the generated locality of the cascade feature and resizing to a ratio of the nose feature size.

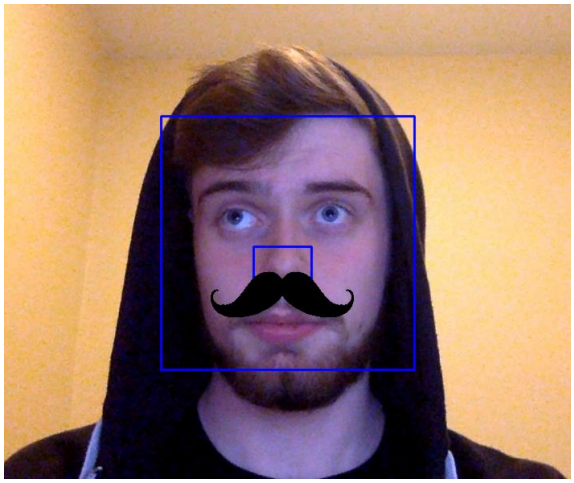

Figure 6 Non-skewed moustache on a Haar-feature alignment (PngImg, n.d.)

Facial Landmark Detection

As pure Haar-feature detection left a lot to be desired, I started using DLIB to annotate and demarcate landmarks on a webcam feed. As described earlier, landmarks positioned on the face allow for the pose to be estimated and various approximate features to be extracted for use. My implementation for positioning items on the face involves taking an anchor position that is approximately constant across face poses, finding the face's pose from these known constants, applying the pose to the flat image through transformations, and then rasterization of the flat image overlay onto the webcam stream.

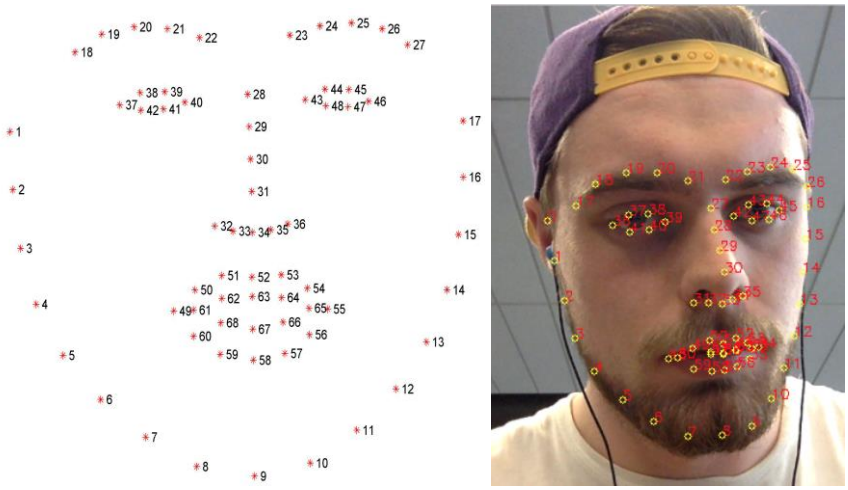

FIGURE 7 ANNOTATION OF KEY LANDMARK POINTS ON THE HUMAN FACE (IMPERIAL COLLEGE LONDON, 2013)

ITEM ANCHORING

Having got the landmarks of the face, I could then determine some constants between frames and poses that could be used to generalise positioning of items. I used three main landmark points for determining the pose – the extreme left jaw-brow position (point 0), extreme right-jaw brow (point 16), and the brow centre point at the top of the nose (point 27). With these in mind, it is possible for me to get the skew of the face, and the size of the face itself. I used these ratios to resize a flat image overlay for the filter object, and skew to the pose of the face per frame.

Adding a moustache builds on this idea by getting the y-value change to the centre-point interpolation between points 33 and 51, applying appropriate ratios for resizing, and then translating to this philtrum position. This method worked quite well as the moustache is skewed well to the face's pose. However, across all filters, there is a small latency per frame, resulting in a laggy experience for the user. A better idea would be to sample every 2 frames to halve the computational expense for the filter applied, or to down-sample the images from the webcam feed to a lower resolution for analysis.

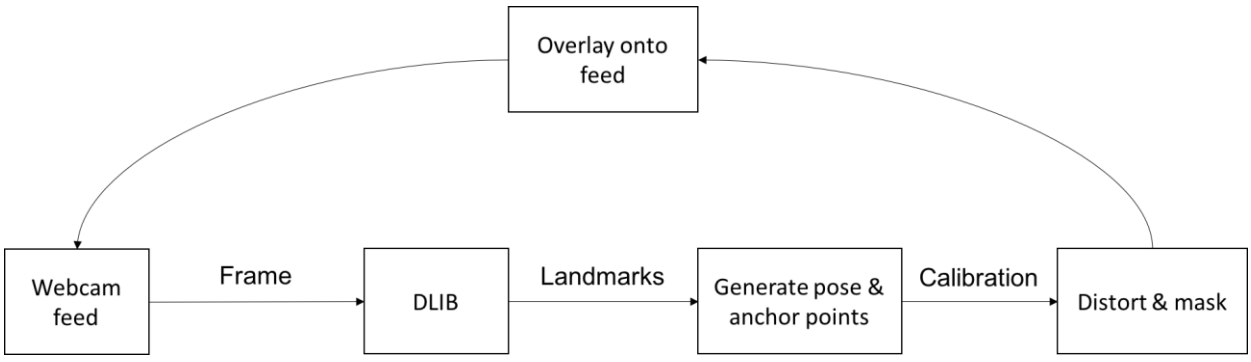

FIGURE 8 LOGIC FLOW FOR THE FILTER OVERLAY SYSTEM

The steps for any flat png overlay can be broken down as follows into these steps:

1. Generate a representation of the face in terms of facial landmarks
2. Using constant points around the eyes, generate a line from one side of the face to the other
3. Calculate the angle of this line and the offset from the centre point between brows
4. Apply this distortion to the overlay and transform to the feature's position
5. In the case of the moustache, generate the appropriate offset from the anchor point between the brows to the new position on the face for the overlay to be placed on the new anchor

Face Swap

The last and most difficult filter applied to the webcam was a face swap method using the Orthogonal Procrustes Problem for the minimisation of matrices. This sequence was broken down into approximately four different steps:

1. Determine the facial landmarks for the image through DLIB
2. Determine the matrix for image B to fit over the matrix for image A
3. Modify the colour balance for image B to match image A
4. Blend and rasterize the section of image B onto image A

Figure 9 Sample face swap (Woodsie TV, 2016)

Having got the features from DLIB, some analysis of the Procrustes issue must be done to allow the image matrices to align up with one another for rotation, translation, skew and scale. The method for determining the positioning of images onto one another can be broken down into the following mathematical steps with Numpy:

1. Cast all matrix inputs into 64-bit floating point matrices
2. Determine the centroid for all landmark points by generating the mean between them per image
3. Subtract the centroids from the relative points per image
4. Determine the standard deviation to remove the scaling issue from the problem
5. Rotate the corresponding image B to A by performing a matrix singular value decomposition
6. Determine and return an affine transformation matrix of the image which generates an appropriate alignment between the image matrices.

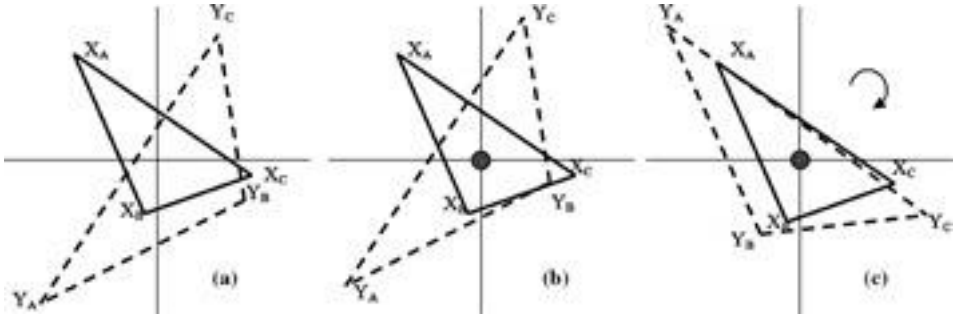

FIGURE 10 ORTHOGONAL PROCRUSTES PROJECTION ALIGNMENT (RESEARCHGATE, 2016)

The next step is to colour correct the portion of image A now overlaid onto image B and fix the tonality of the skin differences and the discontinuities that exist between them. This is achieved by creating a feathered Gaussian blur on the edges of the portion of the images transposed to image B. This results in a crude solution to RGB scale colour correction issues, but this also makes each pixel generate its own scale factor. A problem associated with this is the variations observed when lighting is uneven across the face's kernel, resulting in some blends appearing darker on some sides.

The final step associated with the face swap is blending with a mask and feathering the differences between them in transformations into each other's coordinate spaces through warping and combining elements together in to the final rasterization. The net result is a transfer of facial features.

FIGURE 11 TEST FACE SWAP COURTESY OF PROF. STEPHEN BARRETT (MAANKAAB, 2016)

FIGURE 12 VARYING LIGHTING CONDITIONS WITH A DONALD TRUMP OVERLAY (MAANKAAB, 2016)

Experiments & Results

Glasses & Moustache Filters

Full-Frontal

The flat image filters when applied to the full-frontal face, without any skew, at a large resolution and without any occlusion worked quite well. There was a small amount of jumping between frames due to the slight positional variations, but this could also have been due to noise existing in the webcam feed. The result of this experiment was the filter keeping its correct aspect ratio, and slight variations in position.

Skew, Scale & Rotation

I tilted and skewed my head in multiple ways to give as much variation as possible. The aim of this was to see how well the filter would perform at edge cases where not of the face is not guaranteed to be visible. The result of this test was still positive, where in most cases, the filter was still within an appropriate measure of good aspect ratio, sometimes however with random spikes to an unacceptable skew. Overall, an aggregate average over the past n frames would help to smooth out the position of the skew and noise, but at the cost of latency.

Occlusion

When parts of the image were not visible by another object overlaying part of the face, DLIB was still able to use the remaining points to give a best guess approximation their location. This meant that filters still worked when parts of the face were covered by a hand. This broke down, however, when more than half of the face became occluded. In this case, the filter failed to be applied and the regular feed appeared.

Lighting

I tried a variety of lighting conditions ranging from bright to verging on darkness. On the frame being too dim, the face could not be found correctly, and so no filter appeared. Darkness is a difficult problem in computer vision – variations in brightness can be corrected easily enough, but an entirely dark frame is very difficult to deal with.

Face Swap Filter

Full frontal

A full-frontal image with only one protagonist visible, no skew and no occlusion worked quite well. The series of steps managed to correctly determine the positioning of the transposed matrix onto the image, and the masking and blending steps correctly gauged the tone to an approximate degree.

Skew, Rotation & Scale

The steps began to break down at times with this, as the mask was not appropriately skewed to the face's pose. This could be improved upon, by better calibration of the pose and orientation.

Occlusion

The transposing was correctly performed when part of the face was not visible. The draw-back of this was that the overlay of the new image appeared over the hand occluding part of the face being

rasterized. This created a weird effect where part of the new face appeared above the hand. This is problem could be solved by gauging the field of view and occluding appropriately between z-buffers.

Lighting

With varying lighting conditions, the face was still found and projected accordingly. Issues that occurred with varying light on the face resulted in some strange effects of blotchiness or incorrect dark hues of shadows where one face was projected onto another. This can be fixed, however, by a more careful implementation of blending the faces and masks into one another.

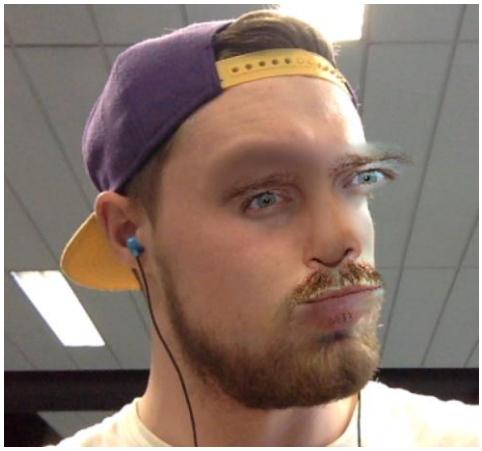

Figure 13 Alignment issue briefly on a frame

Analysis & Discussion of Results

The results of the various filters had an overall positive outcome with the desired basic projections for orienting flat objects and masks onto the facial structure. All three filters satisfy the basic requirements of finding the anchoring points, and then being distorted accordingly to the given pose of the face.

Figure 14 3d perspective of a pair of glasses (VisionExpress, n.d.)

Firstly, the glasses filter is correctly aligned on a regular, full-frontal view of the face without any skew. On analysis, however, given more non-perfect edge cases, the aspect ratio of the glasses distorts slightly on a tilted head, becoming more crushed horizontally. As the pair of glasses encompasses a 3d space, the flat png file fails to show any understanding of depth due to the lack of temples extending from the sides of the glasses themselves to the ears. In a refactor, I would aim to either replace the flat image with a 3d model of glasses and occlude accordingly given some depth information about the view of the webcam, or generate and distort temples depending on the pose and orientation of the head.

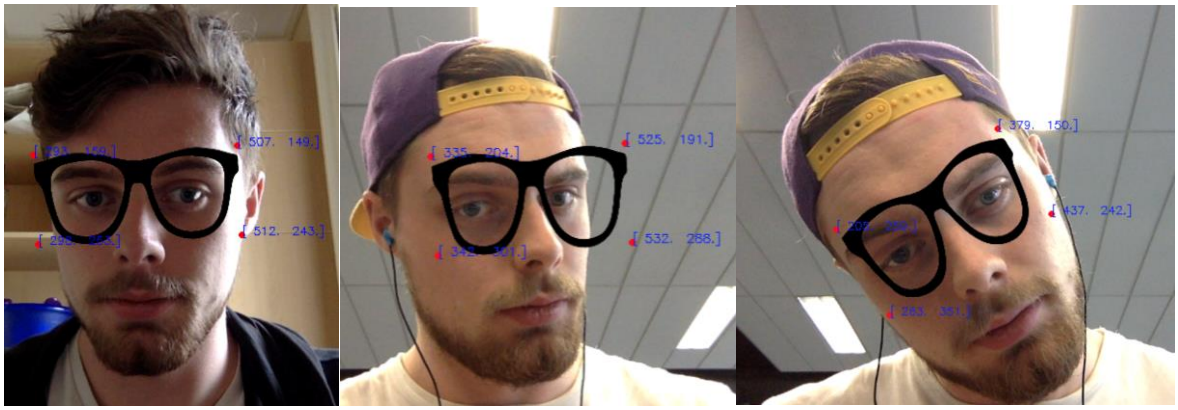

FIGURE 15 GLASSES SKEWED APPROPRIATELY TO THE FACE'S POSE (PNGMART, 2016)

The moustache filter underwent the same distortionary effects of a small change in aspect ratio on tilting the head. As the ratio and position of the flat object is governed by finding the size and angle of the head, both filters suffer from the same distortion as they both undergo this mathematical exercise for finding their anchoring positions. As the moustache is smaller, it is less noticeable; but can still be improved on. Given the smaller set of information about the

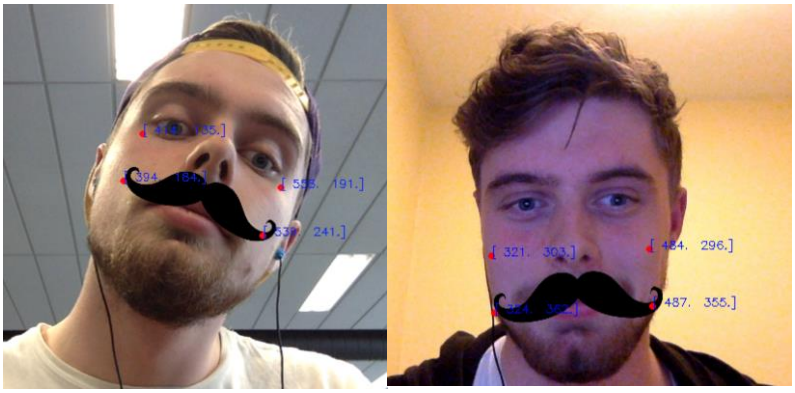

FIGURE 16 ALIGNMENT FROM THE CENTRE-BROW ANCHOR TO THE PHILTRUM (PNGIMG, N.D.)

The final exercise for the filters project was to generate two homographies from facial points of different individuals and to mask one over the other and blend. The result was a slightly disconcerting image where a mask of one image occludes another and has its edges blended to retain skin tone information about its destination.

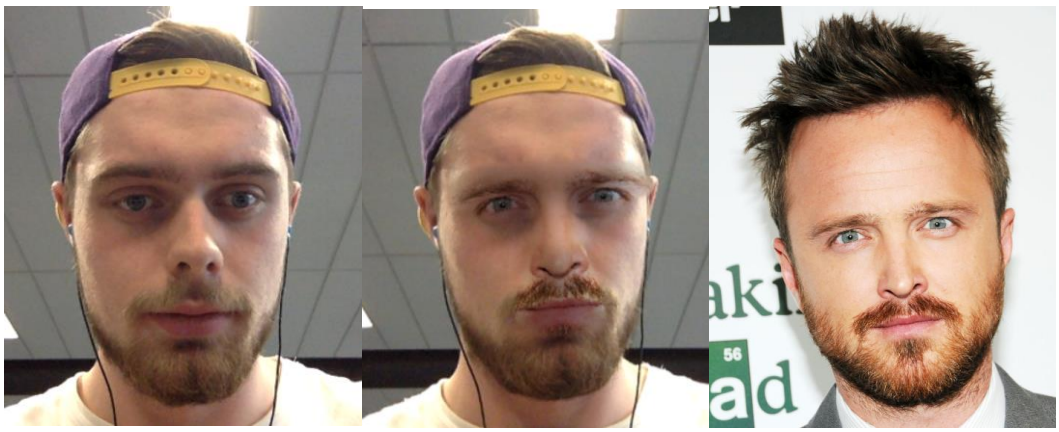

FIGURE 17 FACE SWAP WITH AN IMAGE OF AARON PAUL (HUFFINGTON POST, 2013)

Summary, Conclusions & Future Work

Overall, I was extreme happy with the results of the three filters for this project. The net outcome of the projections, distortions, and masking all went to plan with very interesting features being augmented onto the view of the webcam.

Snapchat filters are fascinating progressions within the scope of augmented reality. The developments that have been applied to the scope of calibration and augmenting of real and fake together into a single application in the palm of one's hand has allowed for interesting deviation of reality to exist. As Snapchat grows in popularity, the quality and variations of the augmenting of what the lens captures expands constantly. In this project, I explored a subset of the theory involved in creating such a filter – spanning over estimating pose through landmark machine learning, augmenting simple flat objects to the pose of the face, and exploring the Orthogonal Procrustes Problem in detail.

While my implementations can be greatly improved on, I believe that it is a great base for future work on for other developers for better and more efficient algorithms for improved latency, better calibration, and rasterizing a three-dimensional object onto a two-dimensional space on a screen, rather than simple a flat image with transparency. I would also recommend for future work to work further on the application of virtual makeup in addition to more filters. Another advancement would be the use of the CLM framework for better pose estimation from landmarks rather than my three-point method of calibrating the pose from simple linear algebra.

BIBLIOGRAPHY

DLIB, n.d. *DLIB.* [Online]
Available at: http://www.dlib.net
[Accessed 18 April 2017].

Earl, M., 2015. *matthewearl.* [Online]
Available at: https://matthewearl.github.io/assets/switching-eds/landmarks.jpg
[Accessed 18 April 2017].

Gower, J. C. & Dijksterhuis, G. B., 2004. *Procrustes Problems (Oxford Statistical Science Series).* 1 ed. s.l.:Oxford University Press.

Huffington Post, 2013. *15 Reasons Aaron Paul Is Awesome.* [Online]
Available at: http://i.huffpost.com/gen/1277268/thumbs/o-AARON-570.jpg?5
[Accessed 18 April 2017].

Imperial College London, 2013. *300 Faces In-the-Wild Challenge.* [Online]
Available at: https://ibug.doc.ic.ac.uk/media/uploads/images/300-w/figure_1_68.jpg
[Accessed 18 April 2017].

Joshi, P., 2015. *http://www.prateekjoshi.com.* [Online]
Available at: https://prateekvjoshi.files.wordpress.com/2014/05/3-pinhole-camera-geometry.png
[Accessed 18 April 2017].

Kazemi, V. & Sullivan, J., 2014. *One Millisecond Face Alignment with an Ensemble of Regression Trees,* s.l.: s.n.

Le, V. et al., 2012. *Interactive Facial Feature Localization,* Urbana: s.n.

Lienhart, R. & Maydt, J., 1998. *An Extended Set of Haar-like Features for Rapid Object Detection,* Santa Clara: Springer.

MaanKaab, 2016. *Donald Trump.* [Online]
Available at: http://www.maankaab.com/wp-content/uploads/2016/11/yeUWaewrR1aPq4DlTAz6_Trump.jpg
[Accessed 18 April 2017].

Manske, M., 2007. *Figure Visually Demonstrating the Evaluation of a Rectangle Feature with the Integral Image,* s.l.: s.n.

Medium, n.d. [Online]
Available at: https://cdn-images-1.medium.com/max/800/1*igEzGcFn-tjZb94j15tCNA.png
[Accessed 18 April 2017].

Mount, J., 2014. *Approximation by orthogonal transform.* [Online]
Available at: http://winvector.github.io/xDrift/orthApprox.pdf
[Accessed 18 April 2017].

OpenCV, 2013. *Camera Calibration and 3D Reconstruction.* [Online]
Available at:
http://docs.opencv.org/2.4.8/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html
[Accessed 18 April 2017].

OpenCV, 2016. *Face Detection using Haar Cascades.* [Online]
Available at: http://docs.opencv.org/trunk/d7/d8b/tutorial_py_face_detection.html
[Accessed 18 April 2017].

Papageorgiou, C. P., Oren, M. & Poggio, T., 1998. *A General Framework for Object Detection,* Cambridge: IEEE.

PngImg, n.d. *PNG images: Beard and moustache.* [Online]
Available at: http://pngimg.com/uploads/beard/beard_PNG6268.png
[Accessed 18 April 2017].

PngMart, 2016. *Glasses Transparent Background.* [Online]
Available at: http://www.pngmart.com/files/1/Glasses-Transparent-Background.png
[Accessed 18 April 2017].

ResearchGate, 2016. *An example of orthogonal Procrustes analysis.* [Online]
Available at:
https://www.researchgate.net/profile/Jesper_Schneider/publication/220434714/figure/fig4/AS:280366825918472@1443856068260/Figure-1-An-example-of-orthogonal-Procrustes-analysis-The-corresponding-points-are.png
[Accessed 18 April 2017].

Viola, P. & Jones, M., 2001. *Rapid object detection using a boosted cascade of simple features,* Seattle: IEEE.

VisionExpress, n.d. *Womens Black Square Glasses.* [Online]
Available at: https://ae01.alicdn.com/kf/HTB11yYpHVXXXXXVXpXXq6xXFXXXL/The-new-ultra-thin-arms-Wire-9352-glasses-frame-glasses-frames-transparent-glasses.jpg
[Accessed 18 April 2017].

Woodsie TV, 2016. *CREEPIEST FACE SWAP EVER!!! | Woodsie has gone too far.* [Online]
Available at: https://www.youtube.com/watch?v=NjA9tBDu9Y8
[Accessed 18 April 2017].