Piecewise linear landmark mapping for pose normalization

Leander Post

June 27, 2020

Abstract

This paper presents two pose normalization techniques based on landmarks and linear mappings between these landmarks. A column based and a polygon based transformation will be discussed and tested with a PCA and an LDA classifier. The results show that the combination of the polygon transformation paired with the LDA classifier gives the best equal error rates. When using PCA, the column and polygon transformations are very close in performance. Overall, both transformations give better scores than leaving the images untouched.

1 Introduction

In the field of facial recognition, pose variation is a common problem most recognizers will have to deal with. Pose normalization algorithms make a synthetic image that looks as if it is taken from a different angle. This way, pose-mismatched images are more comparable. This paper presents two simple piecewise linear mappings to handle the normalization. Both methods involve mapping landmarks from one face onto the other. The first method consist of purely horizontal normalization and the second uses normalization of the features using polygons. The main question will be if these methods can make significant improvements in recognition accuracy over using no normalization. To answer this, faces at different angles will be tested and the scores will be compared.

2 Related work

For frontal view reconstruction-based normalization, according to Chai et al [1] there are two interesting directions researchers take. The first direction is 3D pose estimation, where the 2D image is mapped onto a 3D model. This model can then be viewed from any angle in $\mathbb{R}^3$, and can be projected back onto a 2D image. The homography based normalizer made by Ding et al [2] is an example of this. This direction deals particularly well with noses, filling in the non-visible area behind it by extrapolating the area around it.

The second direction is learned transformations, like Asthana et al [3] and Haghighat et al [4] use. These methods learn a good transformation, based on a training set, by trying transformations based on the shape of the face. By looking at how close the transformation is to the actual frontal image, the transformation can be improved, until a reliable transformation is learned that covers pose variation by simply knowing from previous transformations what works.

Both approaches have been proven to work well for quite large angles. However, when the pose variation is small, the used methods may be more complex than is needed. This paper presents an alternative: a linear piecewise mapping, based on landmarks, that normalizes faces not based on bulky learned data, but just maps features, and with it the face, from the source (domain) image to a target image. There is no prior knowledge needed to do this, which makes it attractive when little training data is available.

3 Method

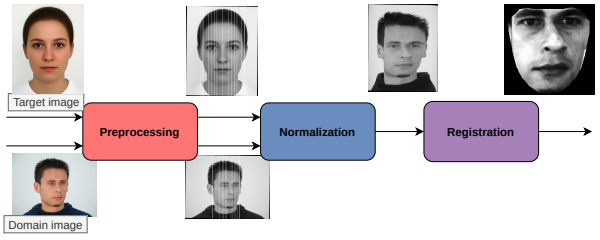

Figure 1: Block diagram of the algorithm

This section describes the preprocessing, the transformations, and the registration, which are performed consecutively, see figure 1. Throughout this

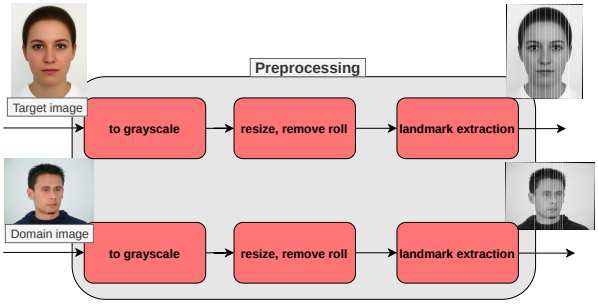

Figure 2: Block diagram of preprocessing

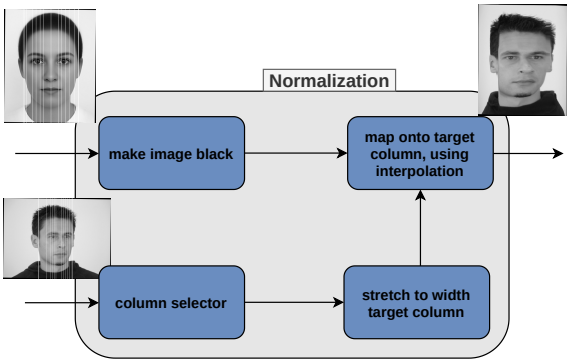

Figure 3: Block diagram of column transformation

paper, the domain image is the image to be transformed, and the target image is the image that is being mapped upon, using the landmarks obtained from the preprocessing.

3.1 Preprocessing

The preprocessing consists of grayscale-conversion, size and tilt normalization, and landmark extraction, in that order, see Figure 2. The first step in the preprocessing is to convert the image to grayscale. This is done to reduce the complexity of the algorithm, and to reduce calculation time. CV2's COLOR_RGB2GRAY[5] method is used to do this, which according to the CV2 documentation uses the following calculation to convert an image from RGB to grayscale:

$$i = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B \qquad (1)$$

Here i is the grayscale intensity, and R,G,B are the red, green and blue intensities, respectively.
After the conversion to grayscale, the image is resized to have a fixed height. This is done to deal with large image sizes, where landmark extraction and transformation get more resource intensive.
For the transformations to work, landmarks on the face are needed. These will be acquired with the Dlib library [6]. It provides 68 landmarks, marking the chin, eyes, eyebrows, nose and mouth. Both transformations, which will be described in the following sections, rely on the landmarks of an average face, called the target from now on. A synthesized image by Dr. Gründl [7] is used for this. The line of best fit is drawn through the eye landmarks. With the tangent of the line, the roll of the face is calculated, and is corrected for by tilting the image the other direction. The angle correction is done with imutils' rotate_bound method[8], which rotates the image while preserving the original image's aspect ratio, without cropping it.

3.2 Column Transformation

The column transformation assumes that the person to be verified is only at a horizontal rotation from the camera. If the face is modeled as a cylinder-like shape, it may be normalized by only horizontal normalization. This means that the face can be sliced up in columns, that are then stretched to fit to the target face's landmarks. In essence, the landmarks are mapped onto each other horizontally, and the image in the column between the landmarks is mapped with it. As Figure 3 shows, the transformation starts by making the target image black, after which a column is selected. This column is transformed to the same width as the target column, and mapped onto the target image. For one of the pixels in the column, the x-coordinate is mapped with:

$$f(x) = (x - x_1) \cdot \frac{x_2' - x_1'}{x_2 - x_1} + x_1' \qquad (2)$$

Here $[x_1, x_2)$ is the interval defining the domain column and $[x_1', x_2')$ is the interval defining the target column.
If this function is used to map the domain onto the range, the output pixels won't have integer coordinates. To avoid this, the range is mapped with the inverse of equation 2, which is obtained by simply switching the positions of x_1 and x_2 with x_1' and x_2'. The coordinates are mapped to the domain image, where most values will also be non-integer. The right intensity to fill into the range-coordinates, is obtained by first order interpolation on the domain image. By doing this for every column, the full horizontal normalization transformation is performed.
The landmarks used to define the columns are defined as a subset of the full set of landmarks, which are ran through an algorithm that ensures that the column coordinates on both the domain and the range are strictly increasing ($x_1 < x_2$ and $x_1' < x_2'$). This is important to avoid the image getting 'folded',

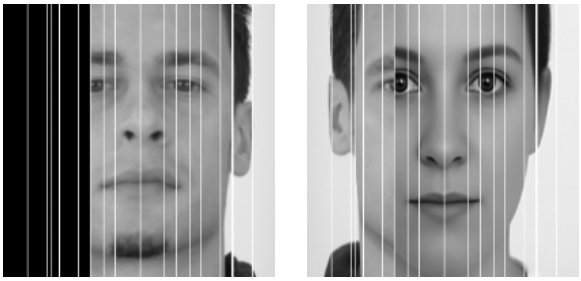

Figure 4: Columns are taken from the domain, get stretched and mapped onto the target

where the same part in the domain gets mapped more than once, causing overlap. Put differently, columns of the domain image, which are cut based on landmarks, are stretched to same width as the corresponding column in the domain. If done from left to right, the new columns can be concatenated to the right, giving the full, transformed image. Figure 4 illustrates this principle.
This transformation will likely work best with small pose variation. In these cases, the cylinder approximation works quite well. The cylinder model is as good as the distance between the facial features and the cylinder. When correcting a larger rotation, the approximation works worse. Also a face that looks less cylindrical will score worse.

3.3 Polygon Transformation

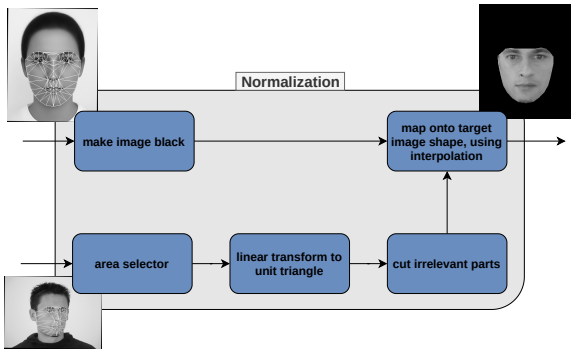

Figure 5: Polygon transformation

The polygon transformation, in contrast to the column transformation, does not assume the head to be cylindrical. Instead it approximates the geometry of the face with a cover of non-overlapping triangular surfaces. By mapping these triangles and their contents onto the triangles of the target image, a non-frontal view can be turned into a frontal one, Figure 6 illustrates this. This transformation maps the triangles one by one onto the target image. The cover of polygons is taken in such a way that most of the landmarks are used, and is inspired by the AAM covers of Asthana [3] and Haghighat[4]. Before mapping the polygons, the mouth of the target image is 'closed', to avoid black pixels. When the domain image has a perfectly closed mouth, there is no information there to be mapped, resulting in black lines. The solution is to take the mouth landmarks to be the average of the top and bottom part of the lips, resulting in a set of landmarks with closed lips. Let's start of with the transformation of the unit right triangle with vertices (0,0),(1,0),(0,1) onto any triangle with vertices (x_1, y_1), (x_2, y_2), (x_3, y_3). Mapping onto $(0,0)$,$(x_2 - x_1, y_2 - y_1)$, $(x_3 - x_1, y_3 - y_1)$ is just multiplying the vector with the matrix:

$$T_1 = \begin{bmatrix} x_2 - x_1 & y_2 - y_1 \\ x_3 - x_1 & y_3 - y_1 \end{bmatrix}$$

To get the points mapped onto the desired triangle, we need to add (x_1, y_1) to all the points. This way, the transform for any point (x,y) is:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = f(x,y)_{pol} = T_1 \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \quad (3)$$

This can be inverted to map onto (0,0),(1,0),(0,1):

$$\begin{bmatrix} x \\ y \end{bmatrix} = f^{-1}(x',y')_{pol} = T_1^{-1} \left(\begin{bmatrix} x' \\ y' \end{bmatrix} - \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \right) \quad (4)$$

This inverse transform maps the pixels in a rectangle surrounding the triangle to a stretched version of it around the origin. We're only interested in points that were inside the triangle to begin with, and those points are located inside or on the unit right triangle after the inverse transform. The points that satisfy this are in the set:

$$A' = \{a \in A | b = f^{-1}(a), b_x + b_y \in [0,1] \wedge b_x, b_y \geq 0\}$$

Here A is the set including all points in the rectangle and b_x, b_y are the x and y-coordinates of $f^{-1}(a)$.
After discarding the points outside the unit right triangle, the triangle can be mapped onto the domain image, to get the pixel coordinates. These coordinates will include non-integers. Interpolation is used to get the intensity value on the domain image at the non-integer coordinates. This value is then filled into the target image. Doing this for a set of triangles covering the entire face without overlap, gives the full transformation.

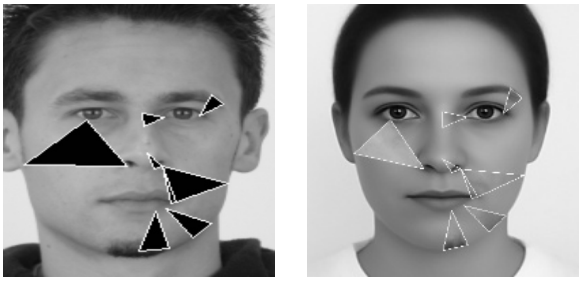

Figure 6: Polygons are taken from the domain, and are mapped onto the target

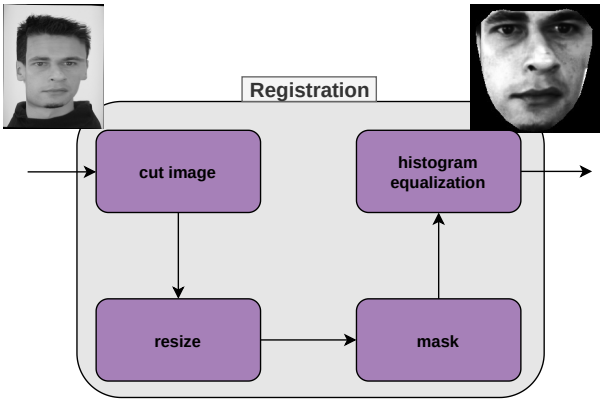

Figure 7: Registration block diagram

3.4 Registration

Three steps are taken in the registration. First, alignment is performed, then for the column transformation, a mask is applied, and then the images have their histogram equalized. Figure 7 shows these steps.

To normalize the faces into comparable images, they should all get the same width and height (w and h, respectively). The eyes should be in the same spot for all images. Additionally, to make up for the stretching of the face by the column transformation, the chin's y-coordinate will be fixed. With the chin and eyes being locked in place, the shape of the face is fixed as well. For both the polygon and column transformation, the information from the landmarks is needed. For the polygon transformation, both x and y-coordinates of the landmarks are those of the target image. For the column transformation, the x-coordinates are those of the target, and the y-coordinates are those from the domain.

The idea is to cut a rectangle, with the relative positions of the eyes and chin constant within those. After the image is cut, the image is stretched and resized into the set dimensions.

To find the values where the image will be cut, the left and right cuts are defined entirely by the eyes. The x position of the left eye is used to fix this location. Because the roll was already corrected in the preprocessing, one can assume that the y-coordinate of both eyes is the same. To define the x-coordinate of the eye, the average of x-coordinates of the left and right eye is used, which will be denoted by x_l and x_r, respectively. If the x-coordinate of the left eye in the registered image is x, the left and right cutting points are $x_{min}$and x_{max}, defined as:

$$\begin{aligned} x_{min} &= x_l - x \cdot r = xl - x\frac{x_r - x_l}{w - 2x} \\ x_{max} &= x_r + x \cdot r = x_r + x\frac{x_r - x_l}{w - 2x} \end{aligned} \tag{5}$$

The fraction, r, denotes the ratio between the distance between the eyes in the uncut image, and the registered image.

For the y-direction, the method is similar. We want to determine y_{min}, y_{max}, the places where the image is going to be cut. For this, we need the y-coordinate of the eyes in the registered image is y, there is also a y_{chin}, the y-coordinate of the registered chin. For y', the y-coordinate of the eyes, the mean of the y-coordinate of the eye-landmarks is used. The coordinate of the chin in the uncut image, y'_{chin}, is the y-coordinate of landmark number 8, which is positioned on the tip of the chin.

$$\begin{aligned} y_{min} &= y' - ry = y' - y\frac{y'_{chin} - y'}{y_{chin} - y} \\ y_{min} &= y' + r(h - y) = y' - (h - y)\frac{y'_{chin} - y'}{y_{chin} - y} \end{aligned} \tag{6}$$

With all values now acquired, the image can be snipped. After this is done, CV2's resize method is used to resize to (w,h). The image is now set to a standard size, but for the column transformation, the background is visible. As the classifier shouldn't be taking the background into account, a mask is applied. To make the mask comparable to the polygon transformation mask, the mask is based on the area that the polygon transformation covers. It is created by polygon mapping a flat image with constant value 1 onto the target image. After resizing, the mask can be applied by simple element-wise multiplication.

To increase contrast, and remove the effect of the illumination on the classifier, the images are also histogram-equalized. This is a well documented function, but in short: a function is defined for the brightness levels of the image that makes the illumination-histogram more spread out, the cumulative distribution is made approximately linear. After the registration, the polygon and column trans-

Figure 8: 11 different poses and the transformations on these poses

form have the same mask applied, and are both high in contrast. Figure 8 shows the original image (top row), the column transformed image (middle row), and the polygon transformed image (bottom row).

4 Experiments and results

All experiments in this report are done on the PUT database. This is a very well controlled database consisting of 100 persons, where all factors except the pose are kept as constant as possible. This gives the classifiers an easy job, but more importantly it makes sure that the performance under pose-variation is tested, and nothing else. The images with horizontal rotation are interesting as a dataset. There are 11 of these images per person.
It should be kept in mind that the goal of the research described in this paper is not about absolute performance. In that respect, there are a lot of improvements that would yield better results. The tests in this section are done to study the characteristics of the different transformations, and should be seen as ways to get a score relative to the other transformations.
To compare the transformations to no transformation at all, the original pictures are also ran through the preprocessing and registration. That means that the original pictures are also stretched to the same head-shape.
For scoring the images, a PCA and an LDA classifier are used on pairs of images, to test the similarity. This way, false match and true match rates can be determined, from which the EER can be determined. Looking at how well the algorithm performs on different angles is interesting, and to quantify this, the EER will be calculated for the different angles available in the data-set. To test the dependence on the choice of training set, the EER will be measured for randomized training/testing splits.

4.1 Classifier

To test the different transformation algorithms, a verification process will be used, that classifies two images as being the same, or being different, based on a distance in some n-dimensional feature space. To this end, two different dimensionality reductions are used. First is principal component analysis (PCA), and second is linear discriminant analysis (LDA). PCA looks at which features of the set of images have the most variation, and in that sense, give the best way to see differences between images. LDA however, looks at what defines classes of images (persons, denoted by having the same label). It looks at the shape of the average class (assuming Gaussian distributed ellipsoid-clouds that indicate the variance), and takes the dimensions with the least variance, that is the features that are the most stable within a class. Both methods have their advantages, PCA is unsupervised, and is better when all classes don't necessarily have a similar shape. LDA yields better scores when the classes are similar, and when labels are available.
Both methods will be implemented. As the principals of LDA and PCA are well documented, the description in this paper will be brief.

4.1.1 PCA

The PCA classifier uses the eigenvectors of the covariance matrix of the set of image vectors to find the principal components (feature vectors). The eigenvalues of the covariance matrix make it possible to select the most important features. All images are projected onto these features, causing a reduction of dimensionality. In this principal component space, the distances between different images can be measured. Using a simple Euclidean measure, the similarity in images can be found. This method is similar to the classic 'eigenfaces' approach, which is described by Turk [9].

4.1.2 LDA

The LDA classifier starts with dimensionality reduction using PCA, which removes noise mostly. It then gathers the classes (linear combinations of features with the same label, thus being the same person), and subtracts the mean of the class from all vectors in the class. The set of normalized features now has the shape of the average class. LDA works by finding the orthogonal vectors with the least variance from this set. By projecting onto these vectors, the classes are separated as good as possible.

4.2 ROC curve

By varying the threshold for the Euclidean distance, pairs of images will be classified differently. Ideally, there would be one threshold, where all distances greater than it would be different persons, and all distances smaller would be from the same person.

As can be seen from Figure 9, this is not the case for

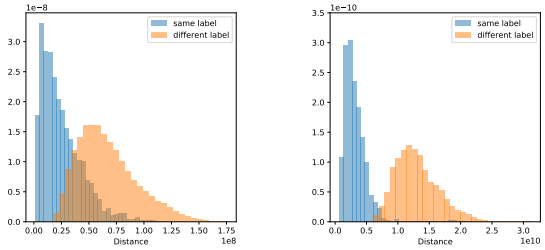

Figure 9: Distance distribution of PCA (right) and LDA (left), both tested on the polygon transformation

both the LDA and PCA classifier. That means there are multiple options for choosing the threshold. To make results more comparable, one can look at the ROC curve (Figure 10). This curve plots the True Match Rate (TMR) against the False Match Rate (FMR), for different thresholds. On the ROC curve, where the false match rate is equal to the false reject rate (1-true match rate), the point called the equal error rate (EER) is positioned. This is a metric that will be used from now on in this section.

4.3 Number of components

The amount of LDA components has a big impact on the score. To choose an optimal setting, the classifier will be tested at different amounts of samples. The results of this can be seen in Figure 11. It appears that beyond 100 components, no significant improvement is made, there is no clear best setting. To keep

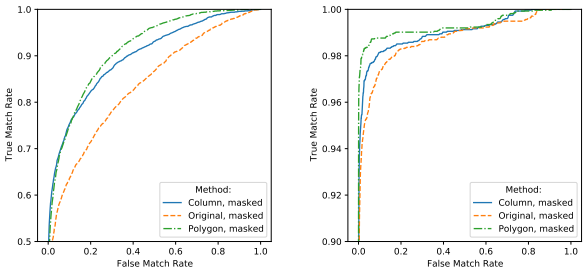

Figure 10: left, right: ROC curve using PCA classifier, using LDA classifier

the number of components relatively low, a default of 100 components was chosen for the remainder of the experiments.

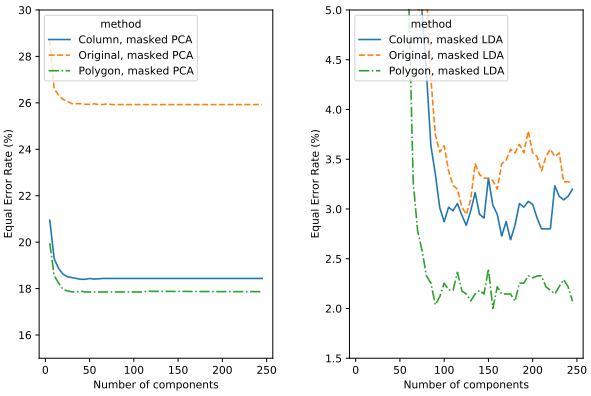

Figure 11: Score of PCA (right) and LDA (left) classifier with varying amount of components

4.4 Training set

Depending on the training set, the results and scores of the algorithms can differ. To quantify this, 100 runs were done for each transformations, at 100 LDA components and 500 PCA components. The means and standard deviations were calculated and are presented in Table 1.

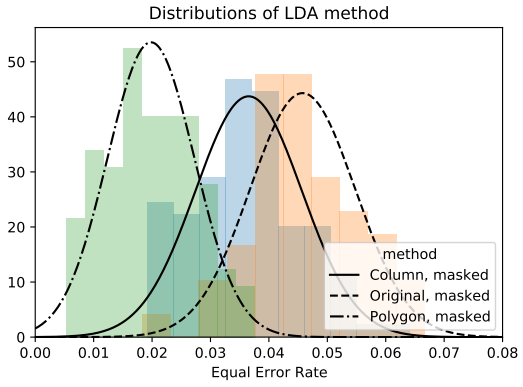

Figure 12: Density plots of the EER of different methods, using LDA on randomized training sets for each run

Figure 12 shows that the split between training and testing set makes a big difference in the outcome. The best score of the polygon transform is around 5 times better than it's worst score.

Method	$\mu(EER)$	$\sigma(EER)$
Column, masked	3.65 %	0.91 %
Original, masked	4.58 %	0.9 %
Polygon, masked	1.98 %	0.75 %

Table 1: EER with different training sets (50 persons), at 100 LDA and 500 PCA components

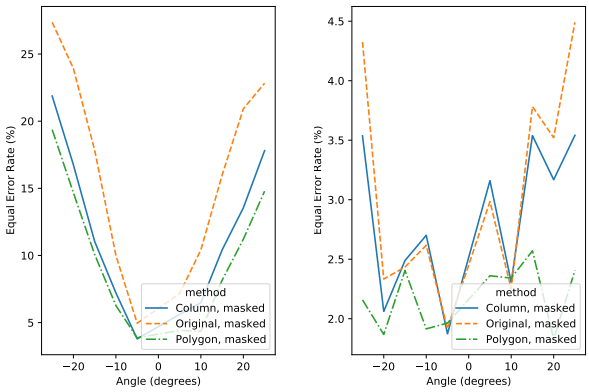

Figure 13: Performance of PCA (right) and LDA (left) under different angles

4.5 Different angles

To test the performance of the transformations under different angles, the algorithm was tested by grouping different angles together, and comparing them to the frontal image.
A 50/50 split on the data set is used, which leaves only 50 possible true matches per angle, making the results jumpy, and very training set dependent. To get a better feel for the performance under angles, the average scores over 100 randomized training/testing splits was taken.
It should be noted that the PUT database has no labels. The two most extreme angles were estimated to be ±25 degrees, see Figure 8, and the angles in between are taken to linearly varying from -25 to 25. The polygon transformation score is quite constant among angles. The column transformation and the original pictures worsen the bigger the angle gets, the column transform does so at a slower rate. The column transformation works considerably better at the higher angles than the non-transformed images.

5 Discussion

This section discusses the results of the previous section, and gives some direction for future work.

5.1 Analysis of the results

The analysis of the results does not follow the structure of the results, as some observations are combined from multiple results.

5.1.1 Loss of shape

Overall, the results show that the polygon transformation outperforms the column transformation in all experiments. This is not a big surprise, as by choosing to map landmarks on top of each other, one decides to use the texture of the face only. When that decision is made, the polygon transformation simply maps the features onto each other a lot better. This can be seen by inspecting the averages of the transformed images. Figure 14 shows that without transforming the images, the features are quite blurry. With the column transformation, the features get better defined, however there is still a lot of blurriness. The polygon transformation, as expected, shows quite a clear image. Especially the mouth and nose clear up a lot. While this gives a nice image, the shape of the nose, chin etc are lost, reducing the differentiability of the classes in that aspect. In the PUT dataset, the structure on most faces is quite comparable. However, when using different camera's, different lighting, people wearing makeup etc, the texture of the skin is greatly changed, making the proposed method in this paper a lot less reliable.

Figure 14: Average faces after no transformation (left), column transformation (middle) and polygon transformation (right)

5.1.2 PCA vs LDA

First of all, it's very clear that the PCA classifier scores a lot worse than the LDA classifier, in every respect. This is probably because of the distance metric on the PCA. Figure 11 shows that the PCA performance stops increasing quickly after reaching around 10 components. This indicates that the first few eigenvectors largely outweigh the other components. This would definitely be something to improve on in the future.
Interesting is the fact that when using the PCA

classifier, the polygon and column transformation are a lot closer to each other than the original pictures, as can be seen in Figure 11 and 13.

5.1.3 Distortion

When looking at the ROC curves, it is noteworthy that both transformations suffer from some images that are very far away from each other in the feature space, while being in the same class, Figure 12 also shows this. This may indicate that there are some artefacts in some transformed images that make it very hard to recognize the imgage properly. It is likely that the big standard deviation on the EER (Table 1) is due to this. When the hard-to-recognize images are included in the training set, they simply can't be classified wrongly anymore, thus increasing the overall score of the algorithm. This causes the score of the classifier to vary greatly between training sets.
The column transformation hasn't performed very well. The cylindrical model of the head doesn't work well for a lot of faces. The more cone-like structure that some heads tend to have, has caused especially the edges of the face to behave in an undesired manner. This is visible in Figure 8, where the background is visible next to the bottom part of the chin.

5.1.4 Performance under angle

The tests ran under different angles show that the column and polygon transformation outperform the original images. While the column transformation might not be very convincing, it does show that it might be worth investigating further in this direction. The polygon transformation seems to perform quite consistently at different angles. It would be interesting to see the transformations performance at larger pose-variation. Interestingly, both the original and column transformation show worse results for +5 degrees compared to -5 degrees. The reason for this could be that the photographed persons tend to look right at the camera for -5 degrees (see Figure 8), and look away at a point

5.2 Future works

In future research, it would be wise to see which images are hard to recognize, and possibly change the transformations in an effort to make them more robust.
Testing the algorithms on other data-sets might point out some issues that need to be improved.
It would be interesting to use non-linear transformations such as splines, to make the mappings smooth, which would follow the cylindrical model better, possibly increasing the quality of the transformations.
At the start of this research, some trapezoid mappings were studied that used the same idea as the current polygon transformation. These could be an interesting new transformation.To get the most out of the transformations, it would be interesting to research the effect of different amounts of components on the score of the classifiers.

6 Conclusion

The goal of this paper was to answer the question whether or not linear piecewise mapping of landmarks (and with it the face) of the sample image onto a target image is an effective way to improve facial recognition under pose variation.
Two methods were chosen to answer this question. One involves mapping columns from the sample image onto the target. The other, slightly more sophisticated approach, maps triangles between landmarks onto the corresponding triangles on the target.
To test the transformations and see the differences in behaviour between the two, classifiers were made to quantify the performance. Overall, the PCA classifier scored worse than the LDA classifier, but both show better results with the column transformation then when nothing is done at all. The polygon transformation scores the best in all of the conducted tests. Using LDA, the polygon transformation got an average equal error rate of 2.0%, the column transformation got an EER at 3.6%, and the non-transformed images showed an EER of 4.6%. All of the methods seem to be quite dependent on the choice of training set, showing standard deviations of at least 0.7%.
One concern with both transformations is the loss of the facial shape. By mapping this way, all faces are morphed into the same (approximate) shape. This can be seen especially in Figure 14. This causes a loss of the facial shape (position of nose, mouth, eyebrows etc), making the algorithm completely reliant on the texture of the face. It is imaginable that this is not enough in some settings, however in controlled settings like the PUT database, the classifiers seem to function quite well. All in all both column and polygon transformation made significant improvements over no transformation.

References

[1] X. Chai, S. Shan, X. Chen, and W. Gao, "Locally linear regression for pose-invariant face recognition", *IEEE Transactions on Image Processing*, vol. 16, pp. 1716–1725, Jul 2007.

[2] C. Ding and D. Tao, "Pose-invariant face recognition with homography-based normalization", *Pattern Recognition*, vol. 66, pp. 144–152, Jun 2017.

[3] A. Asthana, M. J. Jones, T. K. Marks, K. H. Tieu, and R. Goecke, "Pose normalization via learned 2D warping for fully automatic face recognition", in *BMVC 2011 - Proceedings of the British Machine Vision Conference 2011*, British Machine Vision Association, BMVA, 2011.

[4] M. Haghighat, M. Abdel-Mottaleb, and W. Alhalabi, "Fully automatic face normalization and single sample face recognition in unconstrained environments", *Expert Systems with Applications*, vol. 47, pp. 23–34, Apr 2016.

[5] "OpenCV: color conversions", `https://docs.opencv.org/master/de/d25/imgproc_color_conversions.html`.

[6] "Dlib C++ library", `http://dlib.net/`.

[7] P. Fakult, M. Gr, and M. Gründl, "Determinanten physischer Attraktivität – der Einfluss von Durchschnittlichkeit , Symmetrie und sexuellem Dimorphismus auf die Attraktivität von Gesichtern", p. 402, 2011.

[8] "Added 'rotate_bound' method", `https://github.com/jrosebr1/imutils/commit/7cc2522754dbefd8356e0419d1a26de674d4dcde`.

[9] M. Turk and A. Pentland, "Eigenfaces for Recognition", 1991.