STITCHING SMILES

Report
Project A.I.
MSc Artificial Intelligence, University of Amsterdam

Gilles de Hollander
0586544
Gilles.de.Hollander@gmail.com

Thomas van den Berg
5789346
thomas.g.vandenberg@gmail.com

Supervised by:

dr. Theo Gevers
dr. Albert Salah
Gijs Molenaar
Roberto Valenti

February 2nd, 2011

1 INTRODUCTION

The goal of this project was to develop a system that can *automatically* combine a number of group portraits of the same people into a composite where each person "looks good" c.q. smiles. This is a combination of three different topics in computer vision and digital imaging. The first and most mature of these topics is *face detection*, as described in the classical [Viola and Jones, 2001]. Second, there is *smile detection*, i.e. determining how much a given face smiles. Research has been done on smile detection both in videos and still images, examples are [Kotsia and Pitas, 2006] and [Whitehill et al., 2009]. The final topic that our project depends on is *image composition*, the combining of parts of multiple images into a seamless composite. The inspiration for our project was the tool described in [Agarwala et al., 2004] that can compose pictures using an algorithm that computes the minimal cut in graphs. The foundation of this method, using graph cuts for segmentation, was first described in [Boykov and Jolly, 2002].

2 PROBLEM & METHOD

2.1 The Problem

The goal of our project was to design and implement a system that can analyze a series of group portraits of the same people and make a new composition with parts of those group portraits in which every participant smiles. For this the system had to be able to:

1. Find and cluster the same persons in every group portrait,
2. Find, for every person, the "most smiling" face,
3. Stitch the images containing these faces together in such a way that a natural-looking composition of the different group portraits is achieved.

In the coming sections we will describe which approach we have taken to tackle all these problem in sequence and give a quantitative evaluation of some parts of our system and qualitative evaluation of our system as a whole.

2.2 Methodological issues

For our application we had to compose a system consisting of very different components and use them together. There were some methodological difficulties as well as practical problems in the implementation. In the following section we will mainly focus on the methodological difficulties: which algorithms we used, how we have found the right parameters and how we glued it all together. Getting the different parts to work together proved harder than we thought, but we will not describe it in any detail.

3 The System

Alignment

An effective preprocessing step is that the system *aligns* all the input images. In practice this alignment is especially important in large images, where very small movements of the camera can distort the composition of multiple images, even if a tripod is used. When small enough, these shifts are compensated by our alignment method.

To do this, our system takes the first image as a base case and tries to align all the next images to it using an exhaustive search of the local area. It does so by using pointers that correspond to evenly distributed subsampled pixels in the images, and moving these pointers along as it iterates through the different possible shifts. The total difference in RGB pixel values of the pseudo-aligned image and the first image are used as an error measure. A search in a 30x30 pixel window with a subsampling of 1/50 takes only a second or so to perform.

3.1 Step One: Detecting Faces

Detecting faces in still images is pretty much a solved problem, provided that the pose of the head makes for a more or less frontal shot. We have simply used the *Haar cascade detector* implemented in OpenCV, this is more than accurate and fast enough for our purpose. The inner workings of this algorithm, as described in [Viola and Jones, 2001], are not relevant for understanding of our project, but a short explanation of the input and output follows.

As input, we give the algorithm a certain *Haar cascade* to use, these cascades are trained to detect whole faces at different orientations, facial features or certain limbs. We used the "default frontal face" cascade that comes with OpenCV. Furthermore, the algorithm is given a downscaled version of each of the portraits in the series, they are reduced to anything between 0.5 and 2 megapixels. Such an amount of

reduction does not influence the detection rate significantly, but it decreases processing time for obvious reasons.

The output of the algorithm is a list of *rectangles*, formalized as tuples of `(x, y, width, height)`. Overlapping detections, where one face is detected at multiple scales or positions, are merged into one by the OpenCV implementation. False detections do occur, and we filter some of these by removing all faces whose size is more than two standard deviations from the average size, assuming that the people in the group portrait are roughly at the same distance from the camera, and thus their faces are about the same size. This also has the advantage that coincidental bystanders in the background are filtered out. The output rectangles are scaled back to the original size and the image window belonging to each face is extracted.

Clustering Faces

Before we can decide which face is the most smiling for a person, we need to establish which faces belong to the same person. One of our assumptions was that people do not move a lot between different photographs in the series, so we use the position as an indicator of identity. We use agglomerative clustering[1] to cluster all faces that are closer than one "average face size".

3.2 Step Two: Rating Smiles

The next step is very important: the rating of the smiles. To make a composition of multiple group portraits where every person smiles, the system of course has to see who is smiling in which pictures.

To do this the system has a relatively simple, but effective smile classifier that can give a set of faces a continuous 'smile rating', making it easy to rank a number of pictures of the same person by how much this person is smiling.

Preprocessing

To keep the classifier fast enough for real-time use of the entire system, and also to increase its accuracy, some preprocessing had to be done with the face-images that the Viola-Jones algorithm gave as output.

[1] Using `clusterdata()` in MATLAB.

- First, the faces had to be extracted according to the `(x, y, width, height)` rectangles returned by the Viola-Jones algorithm and then further cropped, with the goal to reduce the picture to only just the smile-determining facial features and especially to remove unwanted background noise. We cropped every detected face according to Figure 1.

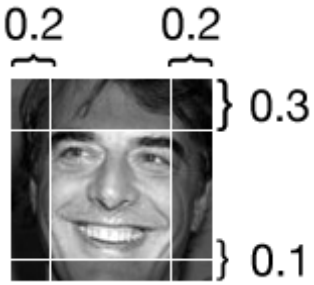

Figure 1: Illustration of the amount of cropping on each side of the Viola-Jones detected faces.

- Next the system equalizes the histograms of the faces. This means that the intensities of every pixel are adjusted in such a way that, as far as possible, every possible intensity occurs with the same frequency, without the ranking in intensity of any pixel being changed. This is done to introduce some illumination invariance to the images that the feature detector and classifier get to process even further.

- Finally, all the images are resized to the same dimensions (64x64 in our standard implementation), these are rather small, as not many pixels are needed for effective classification and it heavily increases the performance of the classifier in terms of speed, which is especially nice in a real-world application.

3.2.1 Edge Orientation Histograms

Our classifier is much-inspired by the review by [Whitehill et al., 2009], in which Whitehill and his colleagues give an overview of different existing face expression recognition-methods and their strengths and weaknesses. He shows that multiple feature/classifier-pairs can give very good results (95%+ accuracy). As our problem is the ranking of a set of faces, which is, we think, simpler than classifying single smiles as either smiling or not-smiling, we

chose the fastest and most easy to set up from this set of well-performing classifiers.

We decided to use *Edge Orientation Histograms (EOH)* features, also called *Histogram of Oriented Gradients(HOG)* in combination with a *Support Vector Machine (SVM)*, as EOHs seemed clear, distinctive features and SVMs known, fast and relatively easy to train classifiers.

In EOHs, every pixel gets assigned a dominant gradient or 'direction' and the magnitude of this gradient. The image is then divided into different areas and for every area a histogram of the dominant gradients is made. Our system divides the image in 7 by 7 (49) blocks, where pixels can have 9 possible orientations. Other settings did not increase performance (they rather slowed the system down severely). These parameter settings meant that the EOH-algorithm would feed the SVM-classifier with 441 features (7 times 7 areas each with 9 orientation bins). Quite a lot features, but no problem for an SVM and far less features than the number of pixels.

3.3 Step Three: Compositing

The output of the smile rating step is a list of the most smiling faces that should be included in the final composite. We construct binary *masks* that indicate the regions from each image that should *surely* be included, these consist of the rectangular face areas corresponding to the most-smiling faces. Obviously, there are pixels that do not belong to any face, and we want to assign these pixels to a source image in an optimal way, in effect forming a 'seam' or 'stitch' between the source images.

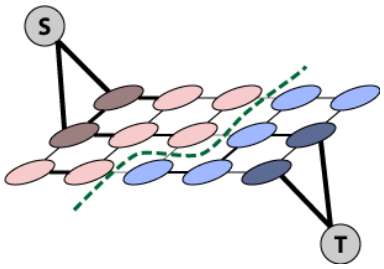

Figure 2: The graph representation of image labeling illustrated. The nodes labeled S and T are the source and sink nodes, connections to these nodes indicate areas that belong to the masks.

The problem of where to allocate this seam can be compared to the problem of finding a *minimum cut* in a graph. A *cut* in a graph is a set of edges that need to be removed in order to completely sever two arbitrary nodes, usually referred to as the "source" and "sink". A *minimum cut* is a cut whose edges have the smallest total weight. By representing the pixels in the image as graph nodes, and connecting each pixel to its (4-connected) neighbors with a certain weight, we can use the efficient min-cut/max-flow algorithm to partition our image. If we base the weights on both images, the minimum cut can represent a seam. In this sense, the weights represent a *cost* of cutting between a certain pair of pixels, illustrated in Figure 2. In our implementation, the weights were assigned as follows. First, the weight of each edge ($W(p,q)$) is simply the sum of the costs of the connected pixels.

$$W(p,q) = c(p) + c(q) \quad (1)$$

$$c(p) = c_{\text{diff}}(p) + c_{\text{grad}}(p) \quad (2)$$

By computing the cost map we can influence the preference of the min-cut algorithm to cut at certain places. Since we want to avoid visible seams, we do not want to cut in places where pixels have different colors in both images, so we give a high cost to places with different RGB pixel values, as shown in Equation 3, where $I_1(p)$ indicates pixel p from image 1.

$$c_{\text{diff}}(p) = ||I_1(p) - I_2(p)||_1 \quad (3)$$

Secondly, in order to add some extra weight to places with mismatching edges, we add the differences in image gradients to the cost map as well, as shown in Equation 4, where $\nabla I_1(p)$ is a 6-vector of the R, G and B gradients in both directions of image 1.

$$c_{\text{diff}}(p) = ||\nabla I_1(p) - \nabla I_2(p)||_1 \quad (4)$$

Multi-image Labeling

A *labeling* is an assignment of pixels to the source images, so that each pixel in the final image is assigned to only *one* of the source images. Using the min-cut algorithm, we can find a labeling based on two source images. By iteratively applying the min-cut algorithm, we can find a labeling that assigns pixels to three or more images.

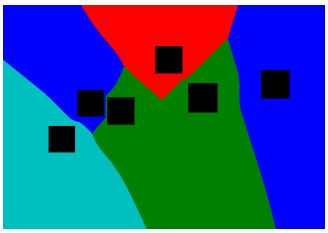

Figure 3: Initial labeling. The black squares indicate faces found by the previous steps.

We start with an initial estimate of the labeling, assigning each pixel to the image in which the closest masked pixel lies (shown in Figure 3). Then we iterate over the labels, similar to the alpha expansion described in [Agarwala et al., 2004]. The min-cut algorithm is applied as if it was a two-label problem, stitching together the current composite, and the source image for the current label α. We force the min-cut step to only assign additional pixels to α by connecting each pixel node that is currently assigned to α to the source node with a high weight. After each cycle through every label we compute the total seam cost by summing the edge weights in Equation 2 for each pair of pixels that have a different label[2]. If this seam cost does not decrease after a full cycle, the loop stops. The final labeling looks like the one shown in Figure 4.

Figure 4: Final labeling overlayed over the composite image.

Blending

Even though the labeling found by the min-cut procedure minimizes visible seams, there are usually still some minor artifacts. Some blending can take care of this. Blending can be done using simple *feathering*, the often used Poisson Image Editing [Pérez et al., 2003], or more advanced methods such as described in [Tao et al., 2010]. We initially planned to use a more advanced method, but by optimizing the seam position only small visible transitions remain and these are effectively masked using *feathering*. When the labeling has been found, binary masks are generated that indicate the complete areas to use from each image. In stead of using these masks directly, we convolve them with a disc shaped *moving average* filter, effectively "blurring" them. The images are then blended proportionately to the soft masks. A sample situation where this is useful is shown in Figure 5.

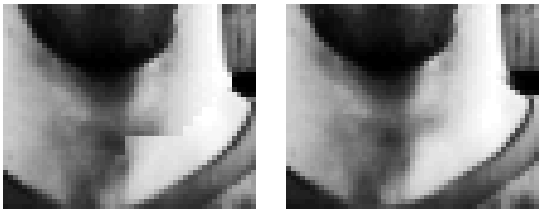

Figure 5: Section of an image showing a seam without and with feather blending, respectively.

4 EXPERIMENTS

4.1 Datasets

For this project we needed some data to test and evaluate. We gathered and labeled one dataset, and we created another by taking photographs.

Smiles

We constructed a dataset of 7,678 pictures of faces, about 3000 of which are smiling. This set is compiled from a number of sources as follows.

- Around 3,000 images from the *Faces in the Wild* dataset [Huang et al., 2007] which we hand-labeled as *smile*, *non-smile* or *discard*[3].
- The Cohn-Canade set contains 250 images labeled with 6 facial expressions. We used the ones labeled "joy" as smiles.
- Around 3,700 images from the GENKI set, developed by and used for [Whitehill et al., 2009], consisting of faces labeled as smile/non-smile, similar to our labeling of *Faces in the Wild*.

We kept aside a random subset of 10% of this data as a validation set, this set was never used for training or parameter optimization.

[2] Refer to [Agarwala et al., 2004], Section 3.

[3] We discarded faces that were occluded and those of which it is not clear whether they smile.

Group Portraits

Because we wanted to test the system as a whole, we also collected a structured dataset of group portraits. This set consists of a total of 127 photos of different groups of people. Each of the 10 group portraits consists of a number of photos of the same people. We set three "difficulty levels". In the *easy* condition, people are standing separately and always look straight at the camera, while they smile in some of the pictures and look neutral in others. For the *medium* condition, we instructed people to move closer together and to turn their heads occasionally. The easy and medium condition are illustrated in Figure 6. Finally, the *hard* condition has the participants moving wildly, perhaps best described as "fooling around", which is a realistic scenario for group portraits.

Figure 6: Typical poses for the easy and medium condition in our group portrait dataset.

4.2 Smile Classification

The first experiment we ran to get a general impression of the accuracy of the classifier, was to train it with different numbers of examples, and then classify some new instances. For each of 5%, 7%, 10%, 25%, 50%, and 100% of the training data, we trained and tested a classifier using 30-fold cross-validation. The resulting accuracy can be seen in Figure 7.

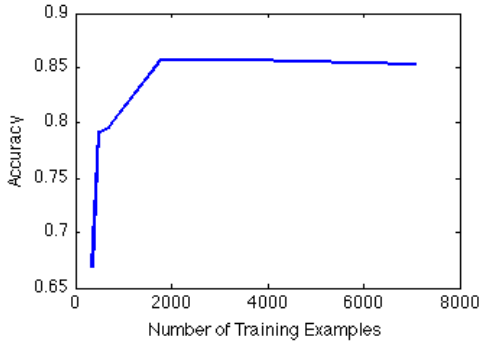

Figure 7: Accuracy vs. training data size using 30-fold cross-validation.

Validation Set Test

Using our final classifier, we classified the samples in the 10% validation set mentioned in Section 4.1. The classification matrix was as follows:

		Predicted		
		Smile	Other	
Class	Smile	107	12	119
	Other	55	405	460
		162	417	579

Yielding an accuracy of 88.4%. The high *specificity* combined with the relaxed constraints mentioned in the next section make this classifier more than powerful enough to perform its task.

4.3 Smile Rating

The context of our systems puts a more relaxed constraint on the classification of smiles. In stead of making an absolute estimation of whether a random person smiles or not, we have three simplifying circumstances:

1. we only need to judge which face smiles *more* than another, relatively.

2. the set of faces to judge belong to the *same person*.

3. even if the order of the ratings is wrong, only the highest rated face has to be a smile.

These conditions allow for a relatively inaccurate algorithm to still perform very well on the task of picking a smiling face. In order to evaluate this ranking we extracted the faces of the people in the group portraits and grouped them by person. For each of the 54 persons, we checked whether the highest ranked face was indeed a smile. In two of the cases the highest rated face was clearly not the most smiling, and in two more it was a face with closed eyes. Since we did not train on open vs. closed eyes, the classifier's lack of accuracy only caused a 3.7% error (2/54) in 'smile picking'. Figure 8 shows a typical set of faces, sorted by their smile rating.

Figure 8: Typical set of faces with their respective smile ratings, as returned by our classifier.

4.4 System as a Whole

It would be very hard to give a meaningful quantitative evaluation. Instead we have tried to give a good qualitative evaluation of the system as a whole by inspecting its results: compositions from different group portrait sets. We tried to take into account different variations in source material. We found that most images in which the people are reasonably static are stitched remarkably well, an example is shown in Figure 9. The system is most easily disrupted by moving people, as in Figure 10.

Figure 9: Example of well-stitched image.

Figure 10: In this image, the participants moved around a lot and subsequently the system fails.

5 CONCLUSION

We have shown that it is feasible to build a system that automatically composes group pictures to include everyone's most smiling face. Not only does it work well in our controlled circumstances, it is fast enough to allow for extensions that increase robustness. It is also useful and fun to use in a real-life setting with real users.
It works under controlled, but still realistic scenarios, where people don't move around *too* much. The results often look almost completely natural and they could replace manually edited pictures, or serve as a preview when people are just 'playing around'.

5.1 Future Work & Discussion

Of course the system is not perfect after just 4 weeks of work. A few aspects of the system should and can be improved on.
First of all, the system assumes the camera does not move: the pictures should ideally be taken with a camera placed on a tripod for the system to work optimally. Small movements can be overcome by the alignment-step, but this part of the system could probably be improved on by using more-advanced alignment-algorithms, based on more complex linear projections.
Second, the system assumes that people are mostly stationary. If they do move the clustering can fail and this can result in very weird and unnatural effects, like the same people occurring multiple times in the composition. An important reason for this is that people are clustered over multiple images by spatial features only: the system looks for the same persons at the same place. This part of the system could probably be heavily improved on by using slightly more advanced features for clustering, like facial recognition: for example a system using eigenfaces can probably perform much better when people move around.
A third property of the system that maybe could be improved on is that it now rates faces for only a very specific feature: whether a face looks in the camera and smiles or not. It could very well be that is not exactly the same as "looking good in a picture". Further research on which features people search for when they assess whether a person is looking good in a picture and how these can be incorporated in an artificial system could be a fruitful way to improve our approach.

References

[Agarwala et al., 2004] Agarwala, A., Dontcheva, M., Agrawala, M., Drucker, S., Colburn, A., Curless, B., Salesin, D., and Cohen, M. (2004). Interactive digital photomontage. *ACM Transactions on Graphics (TOG)*, 23(3):294–302.

[Boykov and Jolly, 2002] Boykov, Y. and Jolly, M. (2002). Interactive graph cuts for optimal boundary & region segmentation of objects in ND images. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 105–112. IEEE.

[Huang et al., 2007] Huang, G., Ramesh, M., Berg, T., and Learned-Miller, E. (2007). Labeled faces in the wild: A database for studying face recognition in unconstrained environments. *University of Massachusetts, Amherst, Technical Report 07*, 49:1.

[Kotsia and Pitas, 2006] Kotsia, I. and Pitas, I. (2006). Facial expression recognition in image sequences using geometric deformation features and support vector machines. *Image Processing, IEEE Transactions on*, 16(1):172–187.

[Pérez et al., 2003] Pérez, P., Gangnet, M., and Blake, A. (2003). Poisson image editing. *ACM Transactions on Graphics*, 22(3):313–318.

[Tao et al., 2010] Tao, M., Johnson, M., and Paris, S. (2010). Error-tolerant Image Compositing. *Computer Vision–ECCV 2010*, pages 31–44.

[Viola and Jones, 2001] Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1. Citeseer.

[Whitehill et al., 2009] Whitehill, J., Littlewort, G., Fasel, I., Bartlett, M., and Movellan, J. (2009). Toward practical smile detection. *IEEE transactions on pattern analysis and machine intelligence*, pages 2106–2111.