EE569 Homework #4

Xiang Gao
9216-3489-10
xianggao@usc.edu

December 1, 2013

1 Face Warping

1.1 Motivation

Face morphing is often used in movies as a kind of powerful spectial effect. The idea is bacically changing one image to another as naturally as possible. To do this task, we should first align several face features in both images by warping. Then we can doing some transformation liking changing colors to change one face to another. In this part, I will try some face morphing techniques.

1.2 Approach

1.2.1 Back to Baby

In this part, we have to select some face features to be used in alignment. Each feature is just a control point which is used in image warping. The more reasonable features we choose, the better the faces are aligned and thus the more natural the transformation turns out to be.

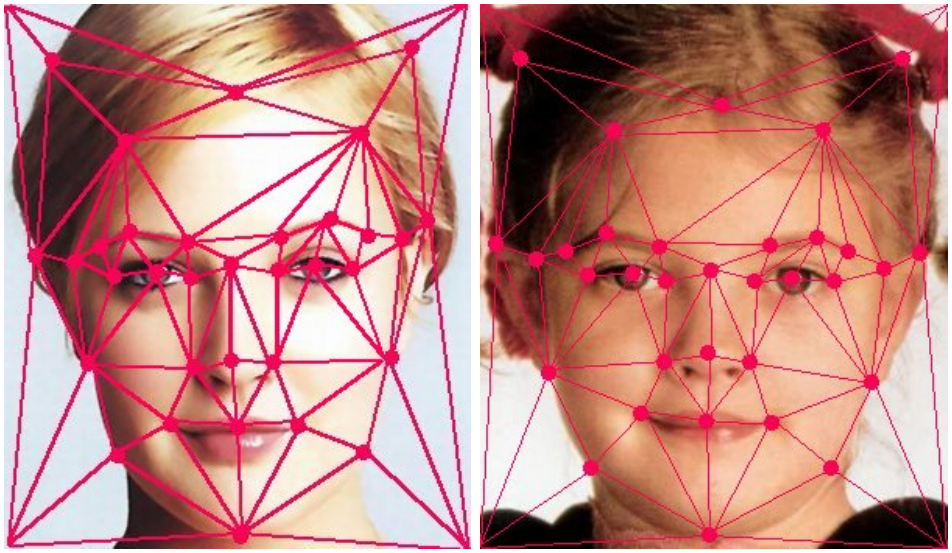


Figure 1: Control points

As is shown above, I used 37 control points for each image and thus 68 triangles that are atoms of warping. The target points are choosing as the mean of each pair of control points. Thus control points in both faces are aligned. I used affine transformation for each triangle which is the same as we used in last homework. The trick here is to determine which triangle each point is belonging to and warping with corresponding transformation matrix.

1.2.2 Face Morphing

After two faces are aligned, we can do cross-dissolving between both images. The equation I used is as follows:

$$I_{out}(i,j) = (1-\alpha) \cdot I_{old\_drew}(i,j) + \alpha \cdot I_{young\_drew}(i,j), 0 \leq \alpha \leq 1$$

1.2.3 Advanced Morphing: From Bruce Banner to Hulk

First I tried to used the same method above, i.e using the same control points and transformation triangles. As expected, the result is not good since the two faces are quite different in both color and the location of features. I tried some other advanced methods. For example change shape while changing color.

Figure 2: Control points

1.3 Results and Discussion

1.3.1 Back to Baby

Figure 3: Face warping with 37 control points

The results shows the warping procedure works fine and there's no large distortion in the image.

1.3.2 Face Morphing

(a) $\alpha = 0.2$ (b) $\alpha = 0.5$ (c) $\alpha = 0.8$

Figure 4: Face morphing

The results shows that two faces are well-aligned using 37 control points. The more control points we use, the better result we can get. We may use more control points to further improve the result. Note that the result is good because these two faces are almost aligned originally.

1.3.3 Advanced Morphing: From Bruce Banner to Hulk

(1)Cross-dissolving Using previous methods with the same control points and triangles:

(a) $\alpha = 0.2$ (b) $\alpha = 0.5$ (c) $\alpha = 0.8$

Figure 5: Face morphing

The drawbacks of this method are obvious. (a)Since control points in both source and target faces are warped to their intermediate positions, these two

faces turn out to be a litte distorted. (b)We only change the color gradually and the morphing seems to be not smooth.

(2)Mesh Warping The first improvement I employed is that for each α, I will change not only color, but also the target control point position. I will warp the source and target images while morphing based on α using follow equation:

$$ControlPoint_{out}(i,j) = (1-\alpha)\cdot ControlPoint_{from}(i,j) + \alpha \cdot ControlPoint_{to}(i,j)$$

This will help us change the shape at the same time morphing color. It outperform the previous method because it's "moving", not "static". For example, the first and last image using this method are exactly the original "bruce" and "hulk" image while they are warped ones using the first method.

(a) $\alpha = 0.2$ (b) $\alpha = 0.5$ (c) $\alpha = 0.8$

Figure 6: Face morphing

However the performance of this method is also based on the control points I choose. For example, in the second image of above results, there is a flaw in the hair. This is because I didn't choose it as a control point. If we want to get a robust method, we have to improve the dependence on control points.

2 Optical Character Recognition

2.1 Motivation

Optical Character Recognition (OCR) is a widely used technique. Like many commercial applications, OCR is used to convert physically printed documents into a digital one and helps us store information digitally. This part we tried to develop a simple OCR system with feature space method.

2.2 Approach

2.2.1 OCR Segmentation and Training

The first step to use a OCR system is to train it. We use a trainning image which contains several clearly printed characters to be used. We have to separate each character. In this part I assume all characters are connected within themselves, thus I can use connected object separation algorithm to do the segmentation. In this part I tried several features first for each character:[1]Normalized Area, [2]Normalized Perimeter, [3] Euler Number, [4] Circularity, [5] Spatial Moment, and [6] Aspect Ratio. Normalized area is the proportion of characters' area and its boundary box's area. However, these features are not good for recognition because characters to be recognised are not very clear thus the symmetry is not a good feature. Based on the features, we generate a decision tree.

2.2.2 OCR Testing

To recognize each symbol in the bill image, we have to do the segmentation first and then recognize symbol by symbol. Note that the "Total amount" in both bills is following the word "TOTAL". Thus if we can recognize a word "TOTAL", then the number following it is the "Total amount" we want to find.

2.2.3 OCR Testing on Restaurant bill

OCR on real-world image is quit difficult. The most difficult part is the preprocessing. There are several reasons. (1) In real-world image there are usually all kinds of symbols. Those symbols may not be within your training set. (2) Charaters in real-world image may not be straight which means you have to choose features that are robust to rotation. In addition, separating symbols from background is a tough task.

2.3 Results and Discussion

2.3.1 OCR Segmentation and Training

After segmentation, I used a random color to represent each segments. The result is shown below:

Figure 7: Segmented characters

The result shows each character is correctly separated. Then we can abstract features from these separated characters.

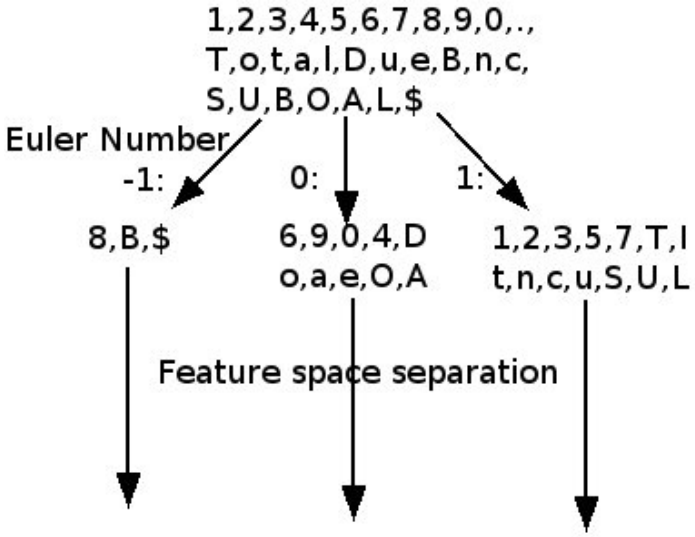
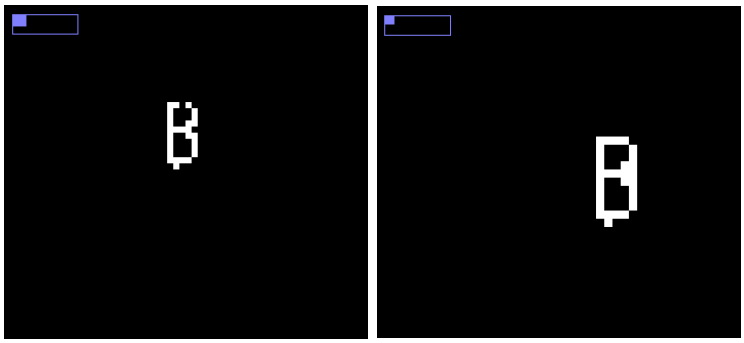


Figure 8: Decision Tree

2.3.2 OCR Testing

A big problem when calcuating Euler Number in test image is that the charater is not fully connected. Thus I tried a simple reconstruction method to reconnect those disconnected parts.



(a) Before reconnecting (b) Reconnected

Figure 9: An example of reconstruction

Now let's see if we can find the word "TOTAL" in the bill. I used the features: [1]Normalized Area, [2]Normalized Perimeter, [3] Euler Number, [4] Circularity,[5] Aspect Ratio in this part and use 1st-order spatial moment as the location of the character. I got part of the result as follows:

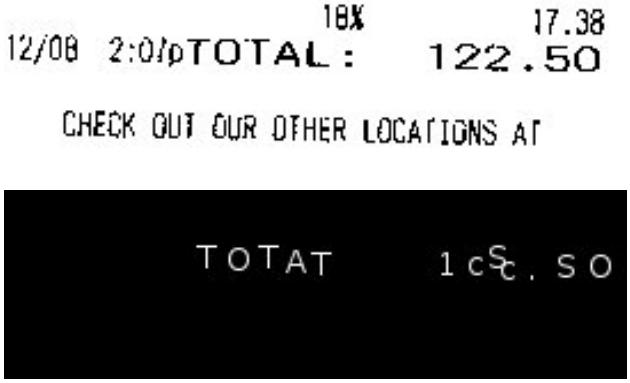


Figure 10: Part of result:test1

We can see the program wrongly detect "L" as "T", "2" as "c", "0" as "O" and "5" as "S". This is because the features I choose can not separate them up.

Figure 11: Part of result:test2

We can see the program wrongly detect “L” as “T”, “4” as “A”, “1” as “L” and “5” as “e”. Note that “5” is detected as “e” because character “5” in the binary image contains a hole, thus its Euler number is 0. According to the decision tree, it’s wrongly recognized.

The features I used do not work well. So I tried to use other features. I add 4 new features which are the ratios of object’s pixels on the boundary box. An illustration is given as follows:

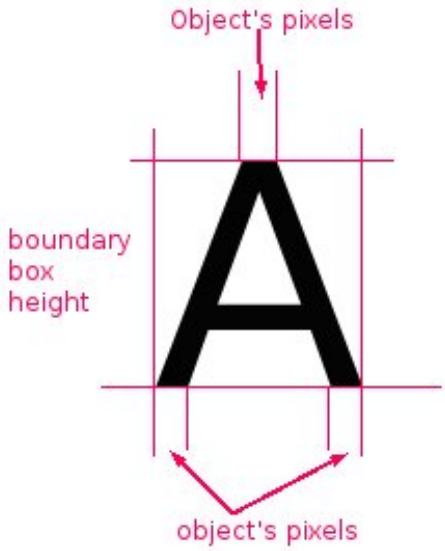


Figure 12: Ratio features

This gives us a new result:

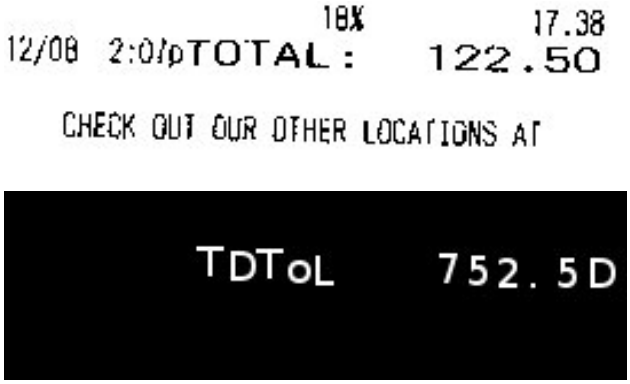


Figure 13: Part of result:test1

However, using these new features, the result of second bill is pretty bad. So these new features are now robust.

2.3.3 OCR Testing on Restaurant bill

When dealing with test images, we faced a problem of binarize the image. Direct binarization with a single threshold is not applicable when the image has shade. I tried an adaptive thresholding method but result is not satisfying.

(a) Threshold = 180 (b) Adaptive thresholding

Figure 14: Binarization

As a preprocessing of OCR, I prefer the first one because the second one contains too many unrelative objects.

Figure 15: Segmentation

A problem in preprocessing is that because of the poor quality of the test image, after binarization many characters are connected. This is a challenge in preprocessing because the segmentation algorithm used in the system can only separate disconnected objects.

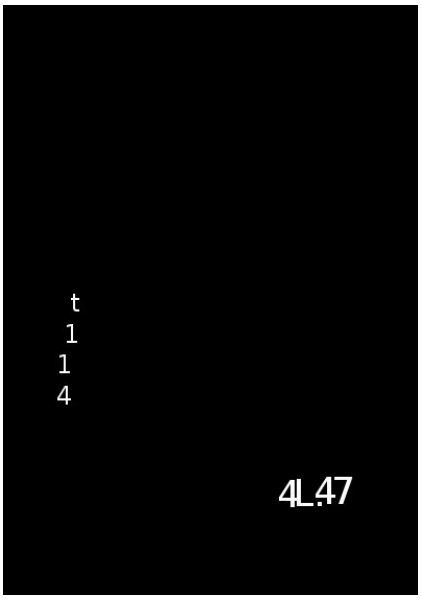


Figure 16: Segmentation

The result is still not satisfying. A "1" is recognised as "t", "6","9" are recognised as "4", "5" is recognised as "7". Note because our decision tree depends a lot on the Euler number. Once the image's quality is too poor to be got Euler number correctly, the result will be of no sense.

3 Image Segmentation

3.1 Motivation

Segmentation is a very important task in computer vision. Good image segmentation gives us a better separation of objects which is useful in object recognition. Mean-shift method is one of the most commonly used method in doing segmentation.

3.2 Approach

3.2.1 Image segmentation using K-means

In this part, I implemented the K-means method to do the segmentation of a color image. Each pixel is corresponding to a single feature point in feature space. By using K-means algorithm, we can get K centroids in the feature space and each pixel in the image can be mapped to a centroid color. Thus we get a segmented image with K segments. First I used RGB channels as three features. Then I tried other additional features to do the segmentation task.

3.2.2 Image segmentation using mean shift filtering[1]

In this part, I implement the mean-shift filtering algorithm with a Gaussian kernel function. The algorithm is described as follows:

1. Initialize j=1 and $y_{i,1} = x_i$.

2. Compute $y_{i,j+1}$ according to $y_{i,j+1} = \frac{\sum_{k=1}^{n} x_k g(\|y_{i,j} - x_k\|^2))}{\sum_{k=1}^{n} g(\|y_{i,j} - x_k\|^2))}$ until convergence.

3. Assign filtered output to be $(x_i^s, y_{i,c}^r)$

where the kernel function we used is Gaussian functions. $g(x) = -k'(x)$ is also Gaussian function.

Using this recursive loop, we will find the local maximum of the probability density whose RGB can be used as outputs. This is the process of mean-shift filtering. An issue that has to be considered is that the computation is huge. Thus I used a searching window to reduce the computation just like that we used in Bilateral filtering.

3.2.3 Image segmentation using mean shift clustering

This part I implemented the algorithm provided in [1]. The method is similar as mean-shift filtering but we have to combine some of its maximums. If two maximums' distance in space is within 'hs' and distance in range is within 'hr', we can combine them to be one color. By doing this, we can get several segments.

3.3 Results and Discussion

3.3.1 Image segmentation using K-means

a) Using RGB as features:

(a) K=2 (b) K=5 (c) K=10

Figure 17: seg1.raw

(a) K=2 (b) K=5 (c) K=10

Figure 18: seg2.raw

(a) K=2 (b) K=5 (c) K=10

Figure 19: seg3.raw

We can see the result of K-means method is not acturally satisfying especially when image becomes more colorful. A drawback of this method is that it's a global method that considers all colors in the image to get centroids. This is not good when doing segmentation where we'd like segments are connected and smooth within each part.

b) I tried using additional features. A first try is using the brightness term Y in YUV color space to help segmenting since within each segment pixels have a similar brightness. However this won't provide any improvement in

the result because Y can be represented by RGB linearly. A second try is to use the RGB channel with the largest value as an additional feature. Note that this is not simply giving more weight to a certain channel. Still this makes no more improvement.

(a) seg1.raw (b) seg2.raw (c) seg3.raw

Figure 20: K=10

3.3.2 Image segmentation using mean shift filtering

a) As a matter of searching window:

(a) windowsize=21 (b) windowsize=41 (c) windowsize=101

Figure 21: seg1.raw

(a) windowsize=21 (b) windowsize=41 (c) windowsize=101

Figure 22: seg2.raw

(a) windowsize=21 (b) windowsize=41 (c) windowsize=101

Figure 23: seg3.raw

A larger searching window may allow you to find a proper local maximum but the tradeoff is the computational complexity. The larger the searching window is, the slower the process will be. Actually there's not a big improvement when we are using 101 than 21 which is much faster.

b) As a matter of spatial radius(hs):hr=16,windowsize=21

(a) hs=1 (b) hs=5 (c) hs=30

Figure 24: seg1.raw

When the spatial radius is very small, each pixel tends to be its own maximum rather than any one of its neighbors. While it grows larger, more and more neighbor pixels are actually involved in consideration. When the spatial radius is much larger than my searching window size, the spatial factor makes not much difference since the spatial term in kernel function is almost flat and the filtered image loses more details which is good for segmentation.

c) As a matter of range radius(hr):hs=16,windowsize=21

(a) hr=1 (b) hr=5 (c) hr=30

Figure 25: seg3.raw

When the range radius becomes large, more and more similar colors join togehter and the better segmentation performance is.

d) A comparison with K-means: Using "seg3.raw" as an example:

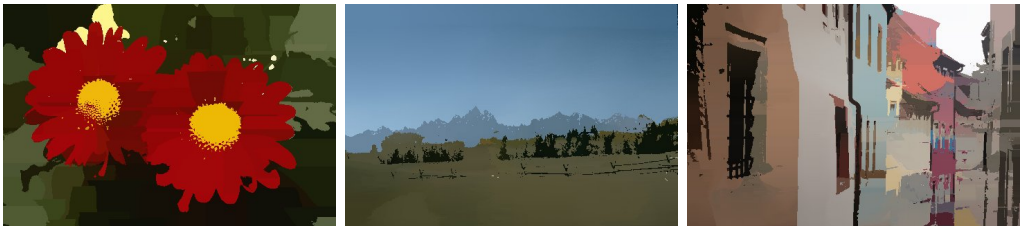
(a) K-means, K=32 (b) Mean-shift,hs=30,hs=30

Figure 26: Comparison between K-means and Mean-shift

Mean-shift method outperforms K-means method mainly because it takes location information into consideration. K-means method is a global method. We can see that the colors change a lot in K-means method because K-means uses global color means. Note that we have only K colors in K-means output. Mean-shift method is a more local method. The number of colors in the output depends on number of maximum. K-means has only one parameter to be adjusted while mean-shift has two.

3.3.3 Image segmentation using mean shift clustering

Searching Window Size = 30, hs = 30, hr = 20



(a) seg1.raw (b) seg2.raw (c) seg3.raw

Figure 27: Mean-shift clustering

The results depend a lot on those parameters you choose like searching window size. Since we combined regions with similar local maximums to be a larger region, there would remain several small regions. Thus the algorithm suggests an improvement by removing those small regions.

References

[1] D. Comaniciu and P. Meer, Mean shift: A robust approach toward feature space analysis, Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 24, no. 5, pp. 603619, 2002.