Table of Contents

EECS Digital image analysis Final report 3

I. My personal views 3

1. Most important thing I learned in this course 3

2. Matlab environment vs. java environment comments 4

3. Course materials/techniques 5

4. Most interesting things I learned 6

5. Field I want to work on 7

II. Project Description 8

1. Goal 8

2. Platform/software 8

III. Design 8

1. The structure for this project 8

2. Why the 3 components are necessary 9

3. Face detection 10

4. Face tracking 13

5. Face recognition 17

IV. Result and analysis 20

1. Result of face detection 20

2. Result of tracking 21

3. Result of recognition 22

V. Remark and future work 24

1. Remark for the final project 24
2. Future work 24
VI. Course feedback and suggestions 26

Liu Liu
Professor Wu Ying
TA Yin Xia
Dec/2013

EECS Digital image analysis Final report

I. My personal views

1. Most important thing I learned in this course

I love to listen to lectures, and by listening to Dr.Wu's lecture, I learned a lot about how to approach a problem rather than using what techniques to solve it. I really like this point because maybe, after ten years, I will not be able to remember all the techniques or tools we were told in class, and that they might be too old for people to use them anymore. The main point is, as long as you know how to come up with the technique or idea, you will always get the problem solved. From what I learned, I can remember the sequence of approaching problems: From a very naïve way of solving the problem, to improving this naïve way to solve more complex questions. I think the most important lesson I learned is never denying a simple naïve answer – because that might be a stepping stone to your next big solution.

However, during lecture, I got distracted sometimes. But there is another way to learn. Like most people, I care about my grades and therefore I learned from the homework assignments, but before that, I need to talk about the environment I used in this class.

2. Matlab environment vs. java environment comments

In the beginning, I was using java as my developing environment because I am most familiar with this language. However, after the first 2 assignments, I feel strongly about the need to learn Matlab because it makes things a lot easier, hence I started picking up Matlab. By reading in the image, you are just reading in a matrix where you can then modify those matrixes using the built-in/custom functions. In java the learning curve for image processing is steeper as the code for reading in different images are different, not to mention that the code for processing different images are different. Comparing to Matlab, java may be more suitable for multiple platforms on cellphones or tablets, but for prototyping and learning new concepts, Matlab is definitely the way to go for image vision, because it provides easy accesses

to all elements in the image.

Another reason for choosing Matlab is because of its real-time debugging system. You can test almost every line of your code by running it in command line. This reduces the large amount of time for debugging and I love this feature. (It is like python command line, just easier).

3. Course materials/techniques

Here is what I learned in the course.

Connected component labeling: This technique is basically scanning through all pixels of an image, and labeling the connected components together. The main difficulty is to find out how to merge same components' labels together, but the basic idea tells us one thing – By scanning through all pixels in an image, we will be able to get all information we need, and thus we can process them. This reduces the run time by avoiding the recursive calls, and made it possible to do CCL on a single computer for large images.

Erosion/Dilation/opening/closing: Those are the basic operations for image

denoising. By Multiplying each block of the image by a convolution kernel, the image will be able to change according to our needs. We did those on black/white images, however, for color images, we can just do this after detection of the objects in the image. The most important concept I learned is using the convolution kernel. With this kernel to manipulate the images, we can do more than just erosion and dilation.

I am skipping histogram equalization because all it do is an averaging/spreading of the current pixel values. Histogram based skin color detection provided an easy way to detect skin color.

4. Most interesting things I learned

I found the canny edge detector and hough transform the most interesting, because they provides new views of how we can approach problems.

Canny edge detector lecture actually provided me a great idea on how to solve problems in the future. Canny used the combination of 2 thresholds to acquire a better result on the edge detection. From this I learned that combining two techniques together is going to provide you with newer ideas and newer ways to

approach a problem.

It is the same with hough transform. By converting traditional coordinate system to angle+distance system, this technique solved the vertical line problem that old coordinate system cannot solve. This provided me a new way of looking at things, to think out side of the box for new answers. Changing to a newer view of the things will help further understating of the problem, and thus will provide many alternative ways to solve it.

5. Field I want to work on

In general, I want to use computer vision as a tool to study human. It seems pretty scary, but this might actually bring profits and make everything nicer and more human friendly. This field has a lot more other applications for us to think about, but for now, we need the technology from which we can base our product on.

Therefore I formed a group of three people to build a basic simple real time face recognition system as our final project. In the future I am expecting to see this system working for recognizing and analyzing human behavior, not just recognizing

a plain face. (FYI this can also be related to studying animal behavior because humans are basically animals wearing clothes)

II. Project Description

1. Goal

Our goal is to find the fundamentals for face recognition, as I mentioned above. We want to find a way to build a real-time face recognition system. We want to combine what we have learned in this class and come up with a more complex application.

2. Platform/software

The chosen programming platform is Matlab 2013a with vision tool box. The reason for choosing Matlab is stated in the personal views/Matlab enviornment, and the vision tool box is equivalent to openCV in java or C++ but under Matlab environment.

III. Design

1. The structure for this project

From the beginning of the class we thought that if we took 1 picture of each of

our classmates, then the system might just work. However, because each person has different poses, we need more information than just 1 picture in our database.

We had a little disagreement on how to implement the project. In order to complete the project, 1 idea is just to input an image to the program and let it decide which person it is. We later on decided to go on with real time face recognition because inputting just an image to the program is not fun at all.

In order to make the system work in real time, we need to have 3 components. The three components are face detection, face tracking, and then lastly face recognition. Those components will be described in detail in the next few sub topics, but we need to figure out why we need those three first.

2. Why the 3 components are necessary

For face detection, it is quite obvious, because if you cannot detect a face, how can you do recognition on it? (Using proof by contradiction). The only problem here is what way is more accurate on finding face.

For face tracking, it is required because we are doing real time analysis. Imaging

you have 30 frames per second, if you do not track a same people how can you do 30 image analyses in a second for finding a face on one computer? This is simply wasting calculation power and therefore not efficient. Hence, face tracking is necessary for avoiding unnecessary calculations.

For face recognition, it is required because the goal of the project is recognizing the face. By comparing the key elements of the face to the faces stored in the database, we will be able to recognize one person, and the difficulty here is still, which method should we choose and how can we make it a real time analysis.

3. Face detection

a. Algorithm description

For this final project, we used vision.CascadeObjectDetector System object in vision toolbox of Matlab 2013a to detect people's face using Viola-Jones algorithm.

Viola-Jones algorithm, to put it simply, is an examining box for a sliding window in an image to try to match different dark/light regions so it can identify a face. The size of the window varies on different scales for different faces, however, the ratio of the window remains unchanged.

Matlab used cascade classifier to quickly determine the regions at which human faces are detected. Each of the cascade classifier consists of stages, and each stage contains a decision stump. According to matlab, they designed those stages to quickly rule out the regions that does not have a face in earlier stages(namely, reject negative samples), and thus save time for analyzing potential region with face in deeper stages.

b. Important code details

Firstly, we need to treat the input source as a video input, therefore we have code

videoFileReader = imaq.VideoDevice('winvideo', 1, 'MJPG_640x480','ROI',[1 1 640 480]);

One thing to notice is that this might not work on any video camera. We chose 640*480 as our input source from the web camera to save memory and processing time.

Then for the cascading Object detector to work, we first created a cascade detector object by calling *faceDetector = vision.CascadeObjectDetector();* as you can see here, we are just using the default vision cascade object detector model without

any input asking for specific or custom trained cascading classifiers. The default cascade classifier will work for detecting frontal faces, but often times for industry standards we need to make custom training cascading classifier.

The next step is to let the vision library to run the Viola-Jones algorithm for detection. We need to see if there is any object being detected in our camera by calling *bbox= step(faceDetector, videoFrame);*

Considering this is a real time analysis, we need to make the machine to check every frame if it did not detect anything by calling the following code:

```
while(size(bbox,1)<1)
    videoFrame= step(videoFileReader);
     bbox= step(faceDetector, videoFrame);
end
```

Stepping the video frame will make sure we keep getting image input from the webcam and by checking the size of the returning bbox we will be able to get the number of face detected in the videoframe.

In order for the next step face tracking to help identify the rotation of the face, we further made a polygon out of the rectangle box to visualize the rotation by calling

x = bbox(1, 1); y = bbox(1, 2); w = bbox(1, 3); h = bbox(1, 4);

bboxPolygon = [x, y, x+w, y, x+w, y+h, x, y+h];

Where bboxPolygon(1:2:end) represents the x values of the polygon and bboxPolygon(2:2:end) represents the y values of the polygon.

4. Face tracking

a. Algorithm description

We are using Kanade-Lucas-Tomasi KLT algorithm to do face tracking. This algorithm is basically based on feature point tracking on the first face, and keeps on tracking it until there is no feature point available.

For the first feature points set on tracking, we used eigenvalue algorithm to find corner points. Basically this is Shi–Tomasi corner detection algorithm which detects the corner. It directly computes the value of eigenvalues to determine whether it is a point of interest or not.

After detecting those points, we will be able to track each of the points we found from Shi-Tomasi corner detection. For each consecutive frame we will try to match the points from the step above. There might be points missing, and if it is the case we will rule them out. As long as there are at least 2 points exist in the videoframe, we will be able to continue tracking the face by finding out the affine transformation of those points.

Then because we are running in real-time, we will want the corner detection algorithm to run again once all corner points are gone.

b. Important code details

We want to detect the points inside of the face region, therefore the following code is going to give us the corner points in face region. *points = detectMinEigenFeatures(rgb2gray(videoFrame), 'ROI', bbox);*

After we got the points, we used PointTracker object in Matlab to track all the points. Then we will not let the face detection algorithm to run unless the detected and transformed points decrease its number to 1. In order to track the points, we

initialize a point tracker object with max biderectional error to reduce noise impact. *pointTracker = vision.PointTracker('MaxBidirectionalError', 2);* Then after initializing the pointTracker with the points locations and videoframe, we will be able to track them by stepping through video frams. *[points, isFound] = step(pointTracker, videoFrame);* Then we need to compare the oldpoints with the new points found. We can get the old points which are still exist in the next frame from oldInliers = oldPoints(isFound, :); and the new locations of the points can be get from *visiblePoints = points(isFound, :);* Notice here that oldInliers have different locations then visiblePoints, and therefore we will be able to find the geomatrix transformation relation between those two by calling: *[xform, oldInliers, visiblePoints] = estimateGeometricTransform(... oldInliers, visiblePoints, 'similarity', 'MaxDistance', 4);* as you can see, we want to figure out the geometric transformation, and we used maxdistance=4 to identify the transformation.

As we got the transformation of the points, we want to use the polygon to track our new location again. Therefore we call *[bboxPolygon(1:2:end),*

bboxPolygon(2:2:end)] ... = transformPointsForward(xform, bboxPolygon(1:2:end), bboxPolygon(2:2:end)); to get the transformed polygon position using the geometrix transformation we acquired by comparing the old inliner locations and the new points locations.

But then, remember we are doing real time analysis. This time we want the algorithm to continue recognizing the face after the first set of points are gone. Therefore we set up the way that if there's no points left on the image, we initialize everything again using the next video frame. If there is no face in the next frame we wait until there is some faces for us to detect and track. Therefore we have *release(pointTracker);* and after resetting everything up we wait until we find a face.

```
while(size(bbox,1)<1)
        videoFrame = step(videoFileReader);
        bbox= step(faceDetector, videoFrame);
        step(videoPlayer, videoFrame);
```

end

5. Face recognition

a. Algorithm description

We used Principal Component Analysis to find a match in the database. This technique is also used in image compression. It is a way of identifying patterns in data, and expresses the data in the way that highlight their similarities and difference.

Basically, first we need a set of data images we store in our program. After we have used PCA on the database, the original data will be in the forms of the eigenvectors we found from the correlation matrix (this is interesting here. Correlation matrix standardized the data).

After inputting an image, we will measure the difference between the eigenvectors in input images with the original images in the dataset, and then we need to determine which picture has the least difference to identify the input image.

One last thing to note, we used face detection in the first step to help create our database in real time.

b. Important code details

In the actual implementation, there are steps for acquiring eigenvectors for database images and input image respectively, and then the comparison will be easily made.

The general eigenvector calculation requires that we subtract the mean of our database.

```
m=uint8(mean(v,2));    %m is the mean of all images, v is the database.

vzm=v-uint8(single(m)*single(O));      % vzm is v with the mean removed.
```

After subtracting the mean from our database set, we will need to find the correlation matrix, and find the eigenvectors to it.

```
L=single(vzm)'*single(vzm);

[V,D]=eig(L);

V=single(vzm)*V;

V=V(:,end:-1:end-(N-1));     %N = 10 which means we are choosing the 10 largest eigenvectors.
```

As you can see here, V can give us the eigenvectors for L, whereas L is the

correlation matrix, ordered by its eigenvalues in D from low to high. Single is just converting data to single precision.

So now we got the largest 10 eigenvectors for our whole database. We want to therefore find the signature for each of our database images. We can simply multiply the eigenvectors by our mean subtracted image to get its signature. *cv(i,:)=single(vzm(:,i))'*V;* where i is the value of different picture and vzm is the database with mean subtracted.

We want to do the same thing for our input image to get its signature, then compare it with each images in the database.

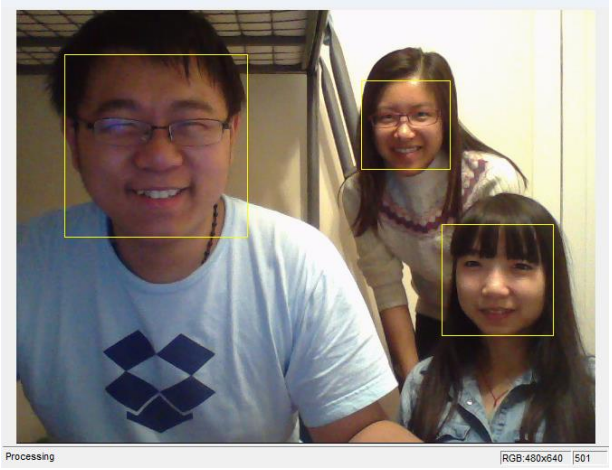
*p=r-m; s=single(p)'*V;* In here r is the input image, m is the main from our database, and V is the eigenvector of our database. We found s to be the signature of the input image.

Then the only thing left to do is comparing the input image signature with what we already have in our database. *z=[z,norm(cv(i,:)-s,2)];* Where z is the matrix containing all differences between input image and the image in database. Later on

we will be able to find the minimal value of z to get the corresponding database image back *[a,i]=min(z); where i is the index of the matched database image.*

IV. Result and analysis

1. Result of face detection



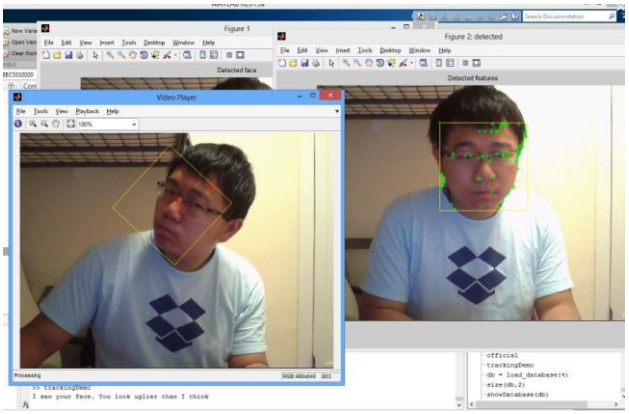
For face detection, our program have >90% rate on detecting the frontal face, thanks to the vision library. However, there are false positives exist, and to further reduce the false positives we need a stronger classifier.

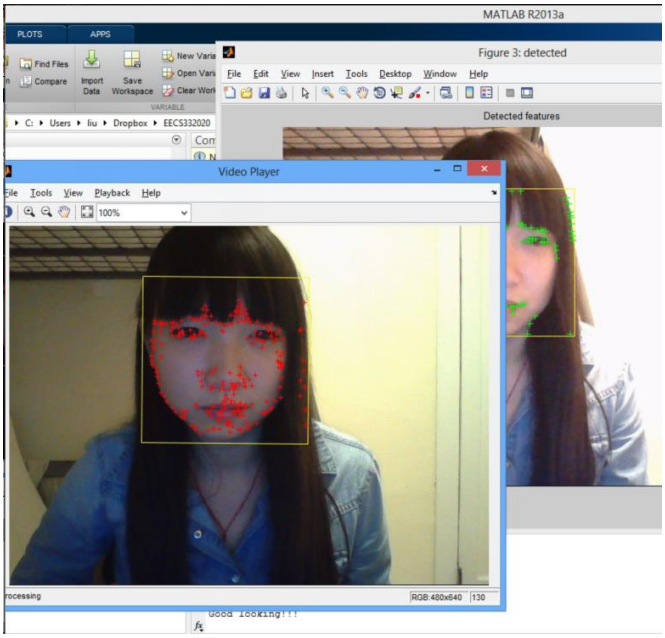
To break this face detection algorithm, a minor issue is to paint a balloon or hold a photo to make the system detect the faces. Those false positives are hard to clean, however, we can integrate other methods to improve the quality like integrating a skin color test.

Another way to break it is through painting the face. If you paint your face so the

dark/light pattern becomes less recognizable, the algorithm fails. This is understandable because even for our human, it is hard to recognize someone with their face painted.

2. Result of tracking





Tracking is working really well on a single target. While rotating the face we can see some corner points missing, and that the polygon window will continue move with our movement when we prevented face detection from running in every frame. The result shows that face detection only needs to be executed when losing the target, and therefore this speeds up our face recognition system.

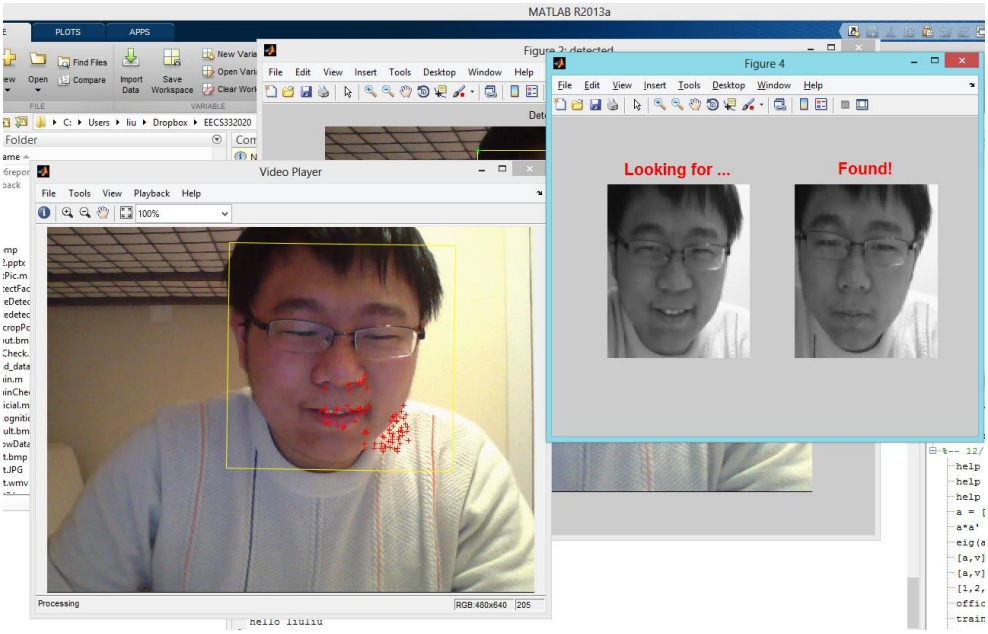
However, we need tracking not only for tracking the same person, reducing the run time of the program; we also need to identify the person's movements in the future.

To break the face tracking program, one just need to make a false positive face detection, and then the tracking algorithm will break because the face detection failed.

Another way to break the system is through lightning. The system lose track of a face if under extreme light circumstances like strong light or very dim light. But still, comparing this with human tracking system, we would not be able to track a person under such circumstance after all.

3. Result of recognition

https://www.youtube.com/watch?v=hYwsnXm0uiw



The recognition rate of our program is about 80% due to the similarities of girls with long hair which covers their identities. Note, this recognition rate does not include the rate of failing the face detection. Therefore, the total success rate for our program is lower.

The face recognition system did not do anything for long rage objects, and for those smaller faces we simply stretch them to make them go under the same procedure as described in the design phase.

Therefore, to break the program, you just have to stand really far away from the camera during training, and then let two girls stand really far away to test our

database. The results will confuse the computer.

V. Remark and future work

1. Remark for the final project

This is a really interesting project to work on. Before I attended this class, face recognition is just a mystical thing and I did not know what to do with it. After this project is completed I suddenly realized I have the ability to create such tools myself. This feeling is great.

However, there is still a long road to go for such technology to be officially implemented in industry standards, because lightning differences, human internal differences (some people like to make up/cover up their true self) all made things a lot more complex than they should be.

Up to this point, face recognition systems are functioning in a normal way, but in order to make it detect faces better, we need future work to be done.

2. Future work

a. What to improve on face detection

To improve the rate of face detection, we need to combine face detection and skin

tone mapping together to make better classifiers. Viola-Johns algorithm will only ensure we have the object with the correct shading, but it does not necessarily be human. Therefore to improve the result, a human skin classifier is necessary.

For those human skin tone classifiers, we need sample collected from all over the world because the skin tones are all different.

b. What to improve on feature detection

Right now we are only using corner detection from the area that we are interested in. This is not entirely accurate due to different lighting conditions and different scales of the faces. In the future, if we have a programmable camera, we will program the camera to get the database sample with the same scale as the real test objects.

c. To improve software interface

Although this is not necessary, we would like to have a software interface that integrated the database creation/deletion/modification as well as real time face detection/recognition together. In that way we will no longer need to type in command line under development mode. This is required for commercial use,

because people are dumb.

VI. Course feedback and suggestions

1. Suggestions

I like the way Dr.Wu perform math problems on the board and by writing them out we will be able to find the mistakes we might make. I would suggest some math problems in the course work to help understanding the materials better.

For example, although I did not use covariance matrix in PCA for the final project, you might teach it in the future. And this requires understanding of calculating standard deviation, variance, covariance, the covariance matrix, eigenvectors and engenvalues. You can start by assigning homework to calculate a 2*2 or 3 * 3 matrix to help understanding those ideas, and then based off that you can keep on talking about how PCA is performed.

2. Final remark

This is a great course overall with lots of materials on the fundamental of computer vision. I learned about how those ideas come from and the linear algebra foundations to the different functions.

As I said in the beginning, the most important thing I learned in this course is how to think in different ways to perfect one solution. The process of acquiring those techniques I learned makes me think a lot more than before, and it may help me develop my own solutions in the future, either in research or in industry. I really appreciate that the lecture taught us about the way of thinking.

Thank you Dr.Wu Ying. Thank you YinXia.