Hi, Thanks that you can go on with this work. This is a document about how to make use of my codes and the remaining difficulties. If you have any questions, please feel free to let me know.

This is my email: [jfunsir@gmail.com](mailto:jfunsir@gmail.com)

## To Do List For Now( From Dr. Sheth)

here is a list of measures that i think we need to implement for analyzing eye movement patterns of movie clips.

1. simple ones: total fixation duration over faces, total fixation duration over eyes, total fixation duration over mouth.

2. still more simple ones: number of saccades made to face regions, to eye regions, to mouth region.

3. so in a movie, all of a sudden a face appears. this happens when a new person suddenly appears in a scene. so now we want to measure the delay in time between when the face suddenly appears and when the person first moves his eyes to look at the new face.

4. if the person moves during the scene, the person's face moves also. we should also measure how well the person tracks the face as it moves. this might be difficult to do but, if we can do it, it wil be very useful. it will be a nice way of finding differences between the two groups of people: normal and abnormal.

5. total duration of fixation, or total amount of time looking at objects other than faces.

6. total amounf ot time spent looking at the movie. so if the eyes are not looking at the movie, then they will be looking elsewhere.

7. also, we should obtain statistics about saccades and fixation durations. like, what is the mean and median fixation duration for a person watching the movie. and how many saccades did each person make while watching the movie. how many long saccades did they make, how many short saccades did they make.

## Get Files

For now you have got the codes and the documents from GITHUB:

<https://github.com/liyunkai/UH_INTERN.git>

But unluckily, the video file and the data file are too big to upload to GITHUB, so you have to get them from the links below:

For video:

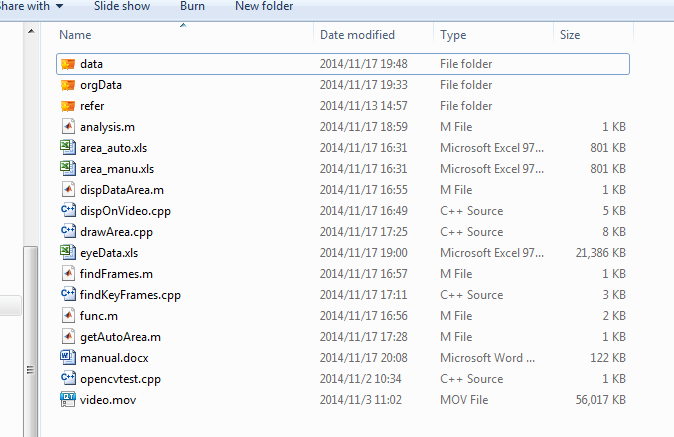
<https://bigfile.bcm.edu/pickup.php?claimID=KoML6eam7BsNzQy1&claimPasscode=jEC0K0SdMK0tNnfc&emailAddr=jfunsir@gmail.com>

For data:

<http://pan.baidu.com/s/1ntG9c9r>

What you need to do is to put the data**\** and the video.mov into the root directory like this:

## Directory structure



Video**\** the root directiry

Data**\** 4 groups of data.

Refer**\** some references from ANI lab.

Video.mov it’s the movie clip which all the data are collected from.

Codes all the codes.

Excels some temp data.

## data

the data are sorted to 4 groups, ASD Oxytocin, ASD Placebo, Controls Oxytocin and Controls Placebo. You can refer to the refer\Subject Groups.xls to get their exact meaning.

Actually the original data is of .tsv format as you can see in the data\orgData directory. The matlab can’t make use of the .tsv file with load method, so I converted them to .xls files. What I have done was opening EXCEL, dragging .tsv files to EXCEL, deleting the first 32 rows then save them as .xls files.

The original .tsv files include more information in their titles, so maybe you will refer to them sometime.

## Video.mov

The movie clip is from an old Hollywood movie Affair to Remember.

There are some issues you should pay attention:

1. The movie clip width and height and their proportion are not fit with the data size. So you need to align them first.

2. The movie clip is fuzzy, and the special area like face, eye can’t be recognized well automatically with the Haar method in openCV. That’s to say we have to draw and the adjust the special areas manually after the automatic process. It looks like a heavy task but actually the movie clip is very short, and I have finished the codes so that you can display your manual result on the video and check them visually. You needn’t worry about that at all.

## Codes Reference Relationship

getAutoArea.m get special area and save them to excel as area\_auto.xls.

drawArea.cpp called as a c++ function to play the video.mov and get the area automatically.

dispDataArea.m load the area\_manu.xls and data and display them on the video.

dispOnVideo.cpp called as a c++ function to play the video.mov and display your manual result on the video to check.

analysis.m load data and area\_manu.xls then draw information you want.

findFrames.m display the sequence of every frame when you play video so that you can get the key frames number.

findKeyFrames.cpp called as a c++ function to do that.

## Step by Step

It looks a little difficult for a beginner to run my codes well, as you see, we need to go on with matlab, C++ as well as openCV. But what we need to do is just to build a development environment, where we can call C++ functions in matlab or call openCV functions in C++. Let’s do it step by step.

#### Software required( You are supposed to have got them installed):

Matlab(2012b) http://www.mathworks.com/products/matlab/

Visual studio(2010) http://www.visualstudio.com/

openCV(2.4.7) http://opencv.org/

#### Call C++ functions with openCV in matlab

##### 0. build development environment

###### 0.1 install compiler

>> mex –setup

* Would you like mex to locate installed compilers [y]/n?

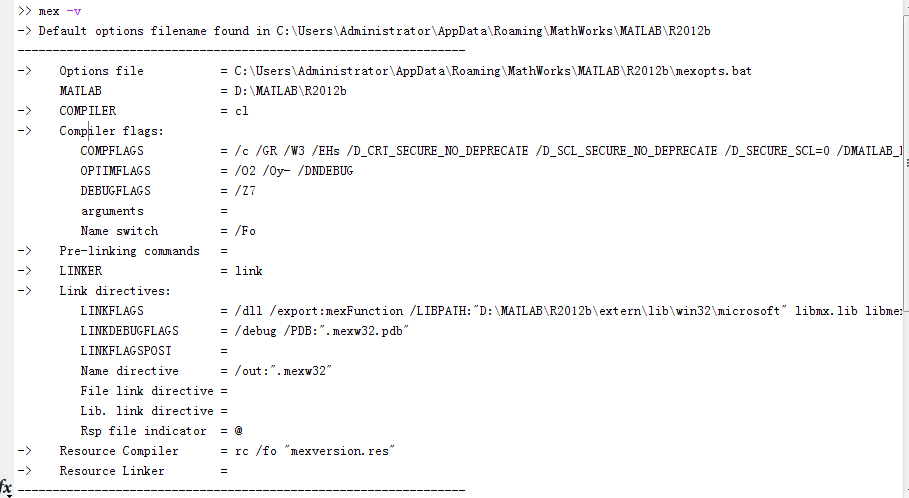
Input y. Then choose the compiler visual studio. It’s the best if you change default compiler to the newest one you have installed.

<http://www.mathworks.com/help/matlab/matlab_external/changing-default-compiler.html>

If there is not the visual studio compilier, you have to Input n and then manually find its installation path to select it.

###### 0.2 build environment to call openCV functions.

>> mex -v



Find the

*-> Options file =*

*C:\Users\Administrator\AppData\Roaming\MathWorks\MATLAB\R2012b\mexopts.bat*

Copy the path to your File Manager, find it and open it with notepad++.

Find the row and add your openCV bin directory path at the tail at below.  
*set PATH=%VCINSTALLDIR%\bin;%VCINSTALLDIR%\VCPackages … %D:\opencv\build\x86\vc10\bin*

Find the row and add your openCV include directory path at the tail as below.

*set* ***INCLUDE=****%VCINSTALLDIR%\INCLUDE; …* ***%;D:\opencv\build\include***

Find the row and add your openCV lib directory path at the tail as below.

*set* ***LIB=****%VCINSTALLDIR%\LIB;%VCINSTALLDIR%\ATLMFC\LIB;%LINKERDIR%\lib;%MATLAB%\extern\lib\win32;%LIB%;****D:\opencv\build\x86\vc10\lib***

Note that you should select the right path. As our system is X86 or X64 and the compiler is different too, you should choose what you are using one.

Find the row and add the files blacked below at the tail.

*set* ***LINKFLAGS****=/dll /export:%ENTRYPOINT … /MAP:"%OUTDIR%%MEX\_NAME%%MEX\_EXT%.map"* ***opencv\_core247.lib opencv\_highgui247.lib opencv\_features2d247.lib opencv\_objdetect247.lib opencv\_imgproc247.lib***

Note that you should make sure of your opencv version and modify the number to the version number as above. For example , ***opencv\_core247.lib*** 247 should be your opencv version number.

Save it and close it.

Now the preparation steps are finished. The next steps are very easy.

##### 1. Compile .cpp file to mex file in matlab.( You must get into the directory path where the codes are )

>> mex drawArea.cpp

##### 2. Call it the same as we call general .m functions.

if you have finished that, you can run the orders in matlab:

>> getAutoArea

The getAutoArea.m will call the drawArea.cpp.

If everything is OK, you can watch two windows of video and the recognized special areas in the right window. If not, please make sure that you have install the compiler in a right way.

There are some links that can help you make sense of the interfaces.

<http://www.mathworks.com/help/matlab/matlab_external/what-you-need-to-build-mex-files.html>

<http://www.mathworks.com/help/matlab/matlab_external/standalone-example.html>

You can find more int the website.

Now you can run the codes very well If everything is OK. If not, you need to check the steps above. If it still can’t work, please email me.

## About excel data

1. the row of strings format will not be loaded in matlab, such as the title row. So you must ensure which is the first row in matlab.

2. The same as I, some columns are also ignored by matlab, so you have to check the mat in matlab carefully.

3. It’s more convenient to convert data that I didn’t attach any titles to the excel data, such as area\_auto.xls.

#### Aera\_auto.xls

1. The data in one row represent several face circle areas in one frame.

2. The data in Column n ( n from 0 to 6 ) represents a circle center X, n+1 represents the center Y, and n+2 represents the circle center radius.

#### Aera\_manu.xls

As you can see, it’s easy to modify the area data in Excel. And you can check the result after you have done this with the dispDataArea.m.

## Code Analysis

When you convert data from Matlab to C++, and vice versa, you must remember the key points:

1. Matlab and C++ story data in memory storage with different ways.

2. The beginning subscripts in arrays of them are different.

In matlab, data in memory storage are stored by column, but in C++, it’s by rows. In matlab, the beginning subscript of array is 1, but in C++ is 0. So you need to convert index when you convert data.

We define that all the area data begin from the first frame to the last of the entire video. The rows number of area data should be total frame number.

There are many informal comments in codes can help you access to them easier.

#### Data From C++ to MatLab (drawArea.cpp)

This is the formula when convert data from C++ to matlab:

typedef struct Area{

double x;

double y;

double r;

}Area, \*pArea;

#define AreaMember (sizeof(Area)/sizeof(double))

#define Columns (AreaMember\*7)

#define ArrSize(frames) ((frames)\*Columns)

#define IndexX(frameSeq,areaSeq,frames) ( (AreaMember\*(areaSeq))\*(frames)+(frameSeq) )

#define IndexY(frameSeq,areaSeq,frames) ( (AreaMember\*(areaSeq)+1)\*(frames)+(frameSeq) )

#define IndexR(frameSeq,areaSeq,frames) ( (AreaMember\*(areaSeq)+2)\*(frames)+(frameSeq) )

First we define the area structure, then we

Define the member numbers as AreaMember

Define the area columns as Columns

Define the ArrSize.

For my area data, they always begin with the first frame of the entire video, so I include the frames total number in the Indexs.

Then I define the IndexX, IndexY, IndexR, which represent the x, y, r of one circle.

#### Data From MatLab to C++ (dispOnVideo.cpp)

typedef struct Area{

double x;

double y;

double r;

}Area, \*pArea;

/\* data Index \*/

#define dataGazeLXColumn 4

#define dataGazeLYColumn 5

#define dataIndexX(frameSeq,startIndex,rows) ( dataGazeLXColumn\*(rows)+(startIndex)+(frameSeq) )

#define dataIndexY(frameSeq,startIndex,rows) ( dataGazeLYColumn\*(rows)+(startIndex)+(frameSeq) )

/\* area index \*/

#define AreaMember (sizeof(Area)/sizeof(double))

#define areaIndexX(frameSeq,areaSeq,rows) ( (AreaMember\*(areaSeq))\*(rows)+(frameSeq) )

#define areaIndexY(frameSeq,areaSeq,rows) ( (AreaMember\*(areaSeq)+1)\*(rows)+(frameSeq) )

#define areaIndexR(frameSeq,areaSeq,rows) ( (AreaMember\*(areaSeq)+2)\*(rows)+(frameSeq) )

#### getAutoArea.m

Set the start time stamp of drawing feature areas.

###### drawArea.cpp

// for here, you need to modify the paths to your local haarxxx.xml paths

const char \*pstrEyeCascadePath = "D:\\opencv\\sources\\data\\haarcascades\\haarcascade\_eye.xml";

const char \*pstrProfileFaceCascadePath = "D:\\opencv\\sources\\data\\haarcascades\\haarcascade\_profileface.xml";

const char \*pstrMouthCascadePath = "D:\\opencv\\sources\\data\\haarcascades\\haarcascade\_mcs\_mouth.xml";

const char \*pstrUpperBodyCascadePath = "D:\\opencv\\sources\\data\\haarcascades\\haarcascade\_upperbody.xml";

const char \*pstrFrontFaceCascadePath = "D:\\opencv\\sources\\data\\haarcascades\\haarcascade\_frontalface\_alt\_tree.xml";

#### dispDataArea.m

###### dispOnVideo.cpp

#### findFrames.m

###### findKeyFrames.cpp

#### analysis.m

###### func.m