Hand segmentation method using multi-frame images

Jilong Wu, Xi He

ABSTRACT

This project is implementing a hand segmentation algorithm that can be used for many applications such as hand recognition. Instead of using a naïve hand segmentation only based on skin color detection, our proposed segmentation method is using multiple frames to assist in obtaining a more accurate hand region. This method will be implemented on a sequence of images (videos). The proposed algorithm is first trained for a set of training videos whose background noise are classified into 4 different levels. In the background whose noise is of the highest level, some non-hand skin regions are included. The training process will provide the set of parameters which produce the best result from training images. In the last step, a set of testing images will be evaluated using the set of optimal parameters. The results of the evaluation are reported. All the output images are compared with a predefined ground truth and the evaluation criteria is based on Dice coefficient.

1. REVIEW OF THE TOPIC

1.1 Introduction

Hand segmentation is the first problem need to be solved in hand recognition problem [3].It is widely studied because of its wide applications in Human Computer Interaction. Many researches have been studied in the field of hand segmentation and many methods have been proposed. Some of them have used skin color detection in the selection process for hand region. Some other methods also mentioned change detection from a sequence of frames of a moving hand. The following papers are selected as references for our proposed method which is doing hand segmentation using two consecutive frames.

In Yoo's paper, it proposed a hand segmentation and tracking algorithm invariant to illuminations from the environment.[1]The binary images with multiple frames are first converted to HSV and then filtered by a color mask. The histogram of the image is updated using the HSV pixel values in the current hand region and the changed region. A histogram back projection is used in the update of the current histogram. In this way, the illumination change in between frames will be weakened. This paper provides a very good start for skin color detection. The target image is first transformed into HSV color. Using HSV color range filtering, only the skin region and regions with similar color to skin from the image are left. This is a very good start to segment hand region since the hard part is located after the color range filtering.

Another interesting paper is from Ghotkar and Kharate's. In Ghotkar and Kharate's paper, three techniques for hand segmentation are explored [2] which are hand segmentation using HSV color space, hand segmentation using lab color space and hand segmentation and tracking algorithm. Also, three different color spaces, RGB, CIE-lab and HSV are discussed. CIE-lab is defined by International Commission on Illumination. The methods discussed in this paper are very time-consuming and the evaluation process is not well discussed. Thus the methods are not appropriate for this project considering the time limit. However, the discussions about implementing HSV color space for hand region detection are be very helpful for this project.

In Li and Tang's paper, it proposes a hand segmentation method combined with skin color detection and motion detection based on complex backgrounds [3]. Unlike the previous paper mentioned before, the segmentation method is based on YCbCr color space. It is very similar to the method we propose in this paper. We also used change detection. Since we haven't understand enough references for YCbCr before, we use HSV instead. However, simple AND operation from change detection as discussed in the paper will not get a satisfactory results.

For segmentation problems, many factors need to be taken into account when determining the quality of the segmentation. For this project, the following issues are considered when evaluating the performance of the algorithm. First, the noise level of the background. We pick four sets of frames with four levels of noise. The noise is defined as how close the color of objects in the background to the color of a skin. Second issue to be considered is moving speed of the hand. If the hand is moving too slow, the algorithm may not detect the change of the hand movement for a given threshold. The third issue to be considered is if there are other skin regions near the hand. Our algorithm is constrained that no other skin regions should be close to the moving hand. Considering the issues mentioned above, our proposed algorithm is discussed in the coming section.

1.2 Overview of remaining sections

In Section 2, the details of our implementation of the algorithm is discussed. It includes the functions we used for each purpose. In Section 3, an experiment design is introduced to adjust the performance of the proposed method. Then the adjusted algorithm is used on some testing images for final evaluation. The last part is the conclusion of this project.

2. PROGRAM IMPLEMENTATION

We implemented the Vision Algorithm in OpenCV and evaluation program using VisionX. The entire application consists of three main parts:

2.1 Application. cpp (main program)

The program is written in C++ using OpenCV and it is the core of the application.

2.1.1 Mode Selection and Overview

This program supports two modes: real-time mode and static mode. In real-time mode, this application runs to implement the algorithm and display the hand segmentation frame by frame captured by an active camera in real-time. In static mode, this application runs to implement the vision algorithm on the frames of the pre-recorded video sets (training/testing sets) and display the hand segmentation of the frames in a sequence/video. This mode is used to train/test the algorithm. When static mode is selected, we can either do training or testing by changing the input video sets in the program. When training, the program reads a sequence of different sets of parameters that need to be tested from a file called testPara.txt and runs the algorithm with each parameter set on all of the training sets/videos one by one. For each set of parameter, the program generates the segmentation (one channel), saves them as videos (store in three Channel though), send them to the ECE server account for evaluation (comparison and calculation of accuracy) by calling the visionTrain.sh file, and then execute the next parameter with the same procedure (note: all the output segmentation sets/videos generated by the previous parameter will be replaced with the current segmentation videos). When testing, the program does the same thing but with only one set of parameter (the ultimate parameter for optimal performance) on the testing video sets and visionTest.sh is called to send segmentation video to the ECE server.

2.1.2 Vision Algorithm

The vision algorithm is applied on each frame of the input. For each frame, it first converts it from RGB to HSV image and apply the HSV threshold on the frame with the HSV threshold parameters to get the thresholded HSV image for hand segmentation. Suppose we have original frame as shown in Figure 1(a). After this step, it gives a hand segmentation but with some noise in the image at same time as is shown in figure 2(a).

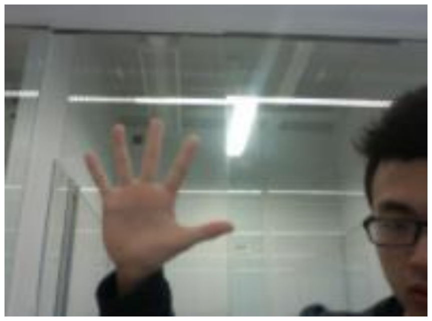


Figure 1(a) Original

Figure 1(b) Luma

Then the morphological filtering is applied. Opening with an ellipse kernel size of 5X5 is used first to remove small objects from the foreground followed by Closing with the same kernel to fill small holes in the foreground. This creates a good result with storing all the regions with hand skin color as shown in Figure 2(b).

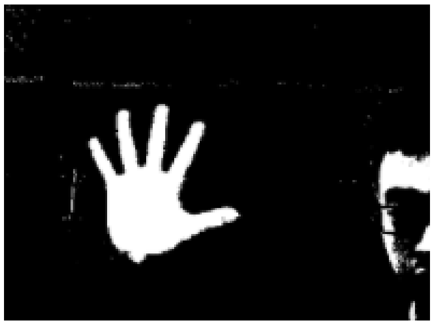


Figure 2(a) HSV threshold

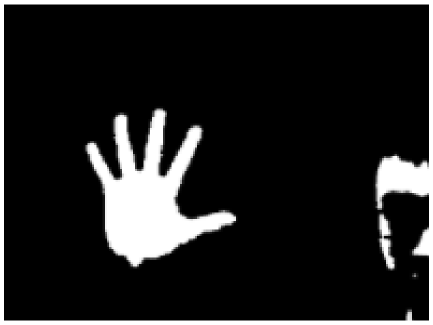


Figure 2(b) Morphological filter applied

In order to get rid of the regions that are not hand but with similar skin color, a further method is used. The program first converts the frame from RGB (3 channel) to Luma(single channel) shown in Figure 1(b). Since the algorithm assumes static background with only moderate hand movement in the frames. it detects the change by comparing each pixel of the frame with the same pixels in the previous frame. If the change of a pixel is bigger than the threshold value defined, it would mark this pixel as a change pixel, specifically 255 if bigger than threshold 0 if not. The change detection frame is shown in Figure 3(a). Next, a AND Bitwise operation is applied on the change frame and the HSV threshold frame to get the hand moved region as is shown in Figure 3(b).

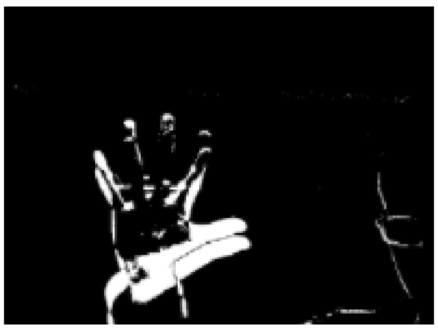


Figure 3(a) Frame change

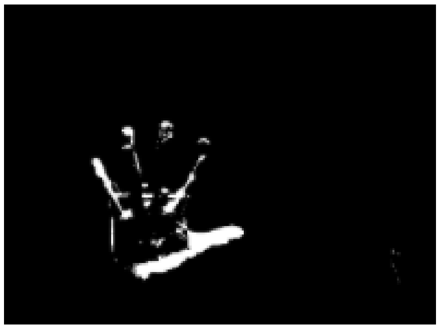


Figure 3(b) AND Operation

This ANDed frame gives a rough contour of the hand shape with lots of broken lines and inaccurate boundary. Since this gives a general clue where the hand is, we use moment to calculate the approximate location of center of mass of the hand using M00, M10, M01 of the ANDed frame. M10/M00, M01/M00 gives the x,y locations of the approximate center of mass of the hand respectively (Note: if M00 is smaller than a certain number, the algorithm assumes there is no hand in the frame since M00 depicts the size of the hand region). The approximate location of center of mass of the ANDed frame is shown in Figure 4(a) red cross. Next a hand region search algorithm is proposed to narrow down the hand segmentation region so that it can find the region of interest to filter out anything outside the region of interest. The region of interest is a square region that is just about to fit the entire hand as shown in Figure 4(a) marked in green. To get this square region, the search algorithm first starts the search in the ANDed image from the hand center location (X,Y) obtained from moment with the search range from (X-m,Y-m) to (X+m,Y+m) where "m" is the search size parameter and it is initialized with 150 to ensure a minimum search region. When searching in the current search region, it checks each pixel location (x,y) with pixel value of 255 and updates the maximum and minimum value of x,y (i.e. column index, row index in image array). If there is an update on any of these values, it increases the search region by incrementing m by 1. Then it keeps doing the same thing and recursively running till there is no more update on any of these values in the latest search region. Then these values are the latest and will be used for defining the region of interest shown in Figure 4(a). Finally after getting the region of interest, the program applies the region of interest on the HSV thresholded image to get the hand segmentation. By doing this, all other irrelevant objects with similar skin color outside the region of interest in the HSV thresholded image will be filtered out shown in Figure 4(b).

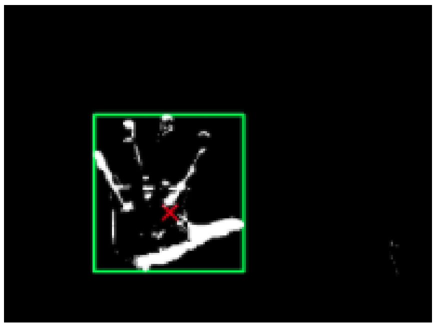


Figure 4(a)Region of interest through search

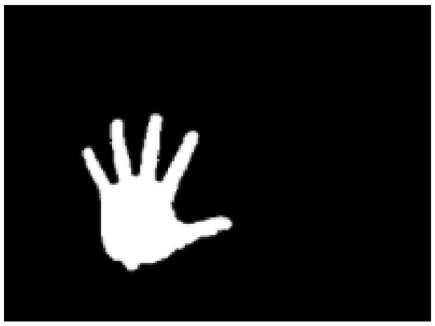


Figure 5(b)Final hand segmentation

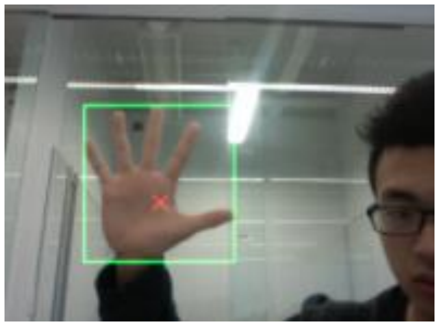


Figure 4(c)Region of interest +original

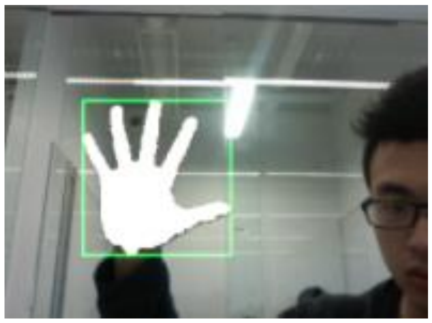


Figure 4(d)Hand segmentation+original

The algorithm flowchart is shown below in Figure 6 and each corresponding images are referred.

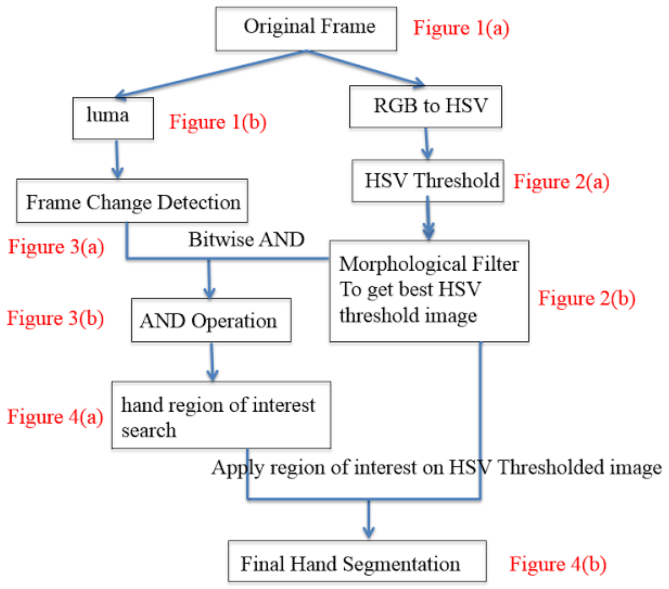


Figure 6 Flowchart of vision algorithm

2.1.3 Core Parameters

Note: For HSV, our normalization is that Hue range is [0,179], Saturation range is [0,255] and Value range is [0,255].

In Yoo's paper: H< 20, 30< S< 150, 80< V<255. We initially used these parameters for thresholding. These parameters are then adjusted to achieve optimal performance in the experiment section. Change Threshold is initially used as 20 and is improved in the experience section.

2.2. visionTest.sh / visionTrain.sh

This script is stored together with the main program. When the main program finishes generating the segmentation result sets/videos for a set of parameter, this script is called to upload the segmentation videos to the server for evaluation. For training, it would upload segmentation video sets after running with each parameter set. For testing, it would upload segmentation video sets for running with the ultimate parameter set only. After uploading the segmentation video sets each time, the script will then login to the server and run a server-side script called EvaluationPara.sh if training mode, EvaluationTest.sh if testing mode to do the evaluation on the segmentation video sets that just uploaded.

2.3 EvaluationPara.sh / EvaluationTest.sh

This script is stored in the server and is called for segmentation evaluation at the user machine remotely after all the segmentation videos are uploaded. EvaluationPara.sh is called by visionTrain.sh for training process while EvaluationTest.sh is called by visionTest.sh for testing. They are very similar except the comparison result is stored in files with different names. In EvaluationPara, using command vmpegtovx -g, the segmentation video is first transformed into .vx format file in gray level and first 30 frames are obtained. Then we clipped and combined the 6th, 11th,16th, 21th and 26th frame for measurement. The last step is to compare those picked frames with predefined ground truth which was labeled by us using VisionX tool. Then use command vrdiff, we get a compare.txt which contains the Dice Coefficient and then we store it in output.txt. EvaluationTest has similar commands except the Dice Coefficient is stored in outputT.txt. We use the data in output.txt and outputT.txt for the evaluation.

The Figure 7 shows the steps of the proposed method:

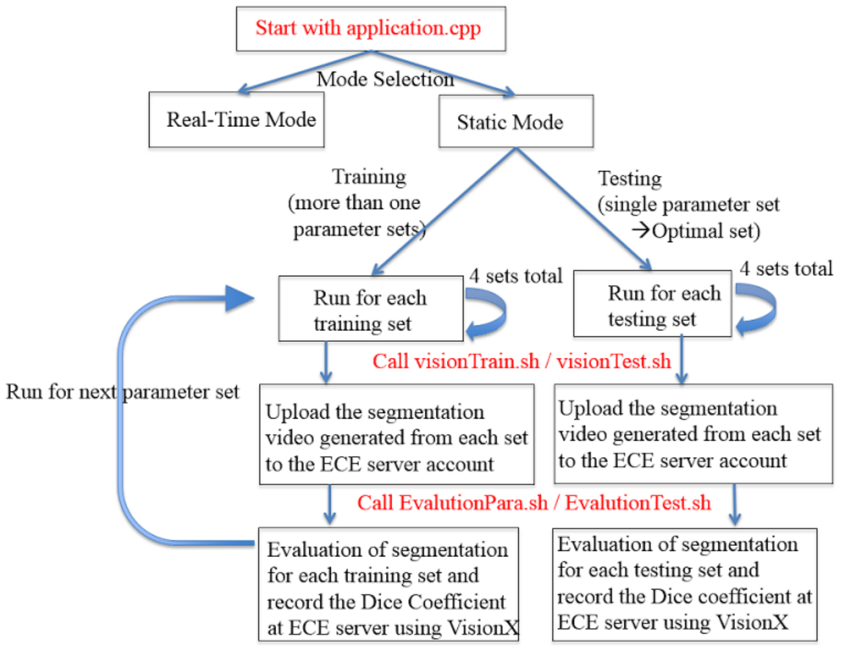

Start with application.cpp
Mode Selection
Real-Time Mode
Static Mode
Training (more than one parameter sets)
Testing (single parameter set →Optimal set)
4 sets total
4 sets total
Run for each training set
Run for each testing set
Call visionTrain.sh / visionTest.sh
Run for next parameter set
Upload the segmentation video generated from each set to the ECE server account
Upload the segmentation video generated from each set to the ECE server account
Call EvalutionPara.sh / EvalutionTest.sh
Evaluation of segmentation for each training set and record the Dice Coefficient at ECE server using VisionX
Evaluation of segmentation for each testing set and record the Dice coefficient at ECE server using VisionX

Figure 7 Flowchart of the proposed method

3. EXPRIEMENT AND EVALUATION

3.1 Experiment design

In general, the experiment has two phases: the first phase is to use some training video sets to train the proposed method with different sets of parameters. Given the same set of training examples, we collect the performance data of each set of parameters (how the performance is measured will be discussed later). Then we plot the performance of each set of parameters and pick the set of parameter which produces the best performance. In the second phase, we have another testing video sets. Using the optimal parameters we obtained in the training phase, we obtain the segmentation performance of the testing sets and evaluate our proposed method.

As mentioned in the last part of Section 1, there are three issues to be considered for the proposed method. Thus, when picking training and testing video sets, these issues need to be considered. Since the method is implemented in multi-frame images, our training sets are collected using short videos (~3s mpeg format). The training sets are classified into four categories based on the background noise level. The noise is defined as how busy the background is. In the level four noise level, there are some other body parts included in the picture. Similarly, the testing sets are also collected based on four categories but with different background settings. The fourth one also has some body regions in it.

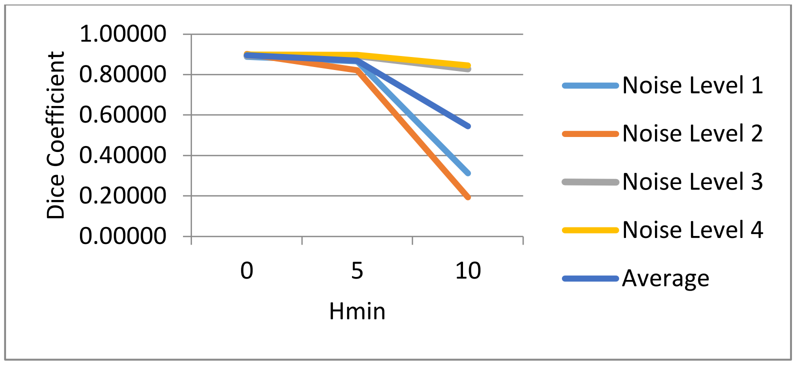
To measure the performance of the result, we compare the output from the proposed method with the predefined ground truth. Then using the command from VisionX, the Sorensen Dice coefficient is calculated. We use Sørensen dice coefficient to represent the accuracy of each frame: $Accuracy = \frac{2|A \cap B|}{|A|+|B|}$, where $|\boldsymbol{A}|$ is the predefined hand area drawn by us and $|\boldsymbol{B}|$ is the hand segmentation result from the designed algorithm/method. $|\boldsymbol{A} \cap \boldsymbol{B}|$ is the overlap area of the predefined area (ground truth) and the resulted segmentation area. $|\boldsymbol{A}|+|\boldsymbol{B}|$ is the sum of the predefined area and segmentation area.

The short training and testing videos are about 3s and with 20+ frames in each. After the videos are fed in the program, they will be divided into frames and the segmentation of each frame will be produced. In the evaluation process, we only predefine the ground truth and compare the segmentation results for the 6th, 11th, 16th, 21th and 26th frames. Using the command in VisionX, we can calculate the Dice coefficient for all these selected frames. For every given parameter, the final accuracy of the proposed method will be the Dice coefficient average of the four different levels of backgrounds.

For our proposed method, since we use HSV color thresholding and frame change detection, the following seven parameters are considered during the training process: the thresholding range of H (Hmin and Hmax), the range of S (Smin and Smax), the range of V (Vmin and Vmax) and the frame change threshold. The range of H, S and V determines how well the skin region can be located and the frame change threshold determines how well the moving object can be captured. The experiment results and observations are reported in the next part.

3.2 Experiment results and observations

As discussed in the experiment design, seven parameters are adjusted to improve the performance of the algorithm. (Note: when one parameter is trained, all other parameters are kept consistent). The training results for each of the seven parameters are shown below. The average in the plot is the Dice coefficient average of the backgrounds of four different noise levels. The table of the comparison data is available in Appendix.



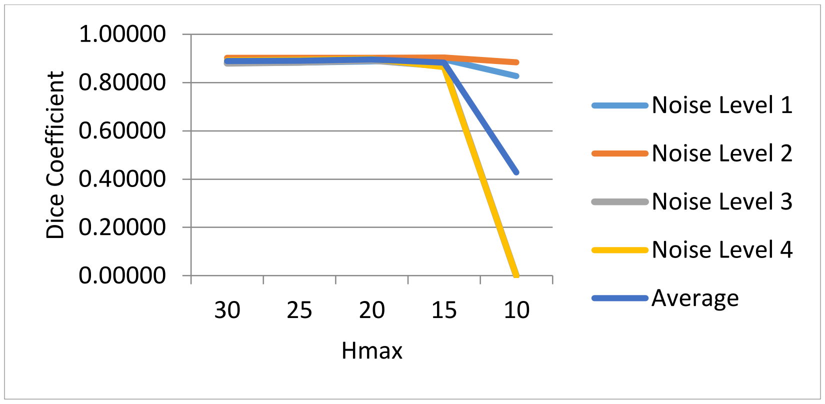
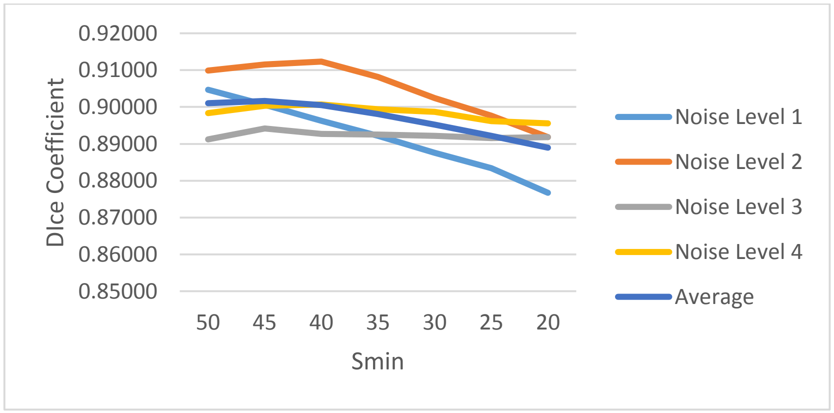


Figure 8 Training results on H values

As we can see in Figure 8, Hmin is the lower threshold for H value. It is trained from 0-10. Hmax is the larger threshold. It is trained from 10-30. As shown in the figure, the optimal value for Hmin is 0 and for Hmax it is 20. They are picked according to the averaged values of the four kinds of backgrounds.



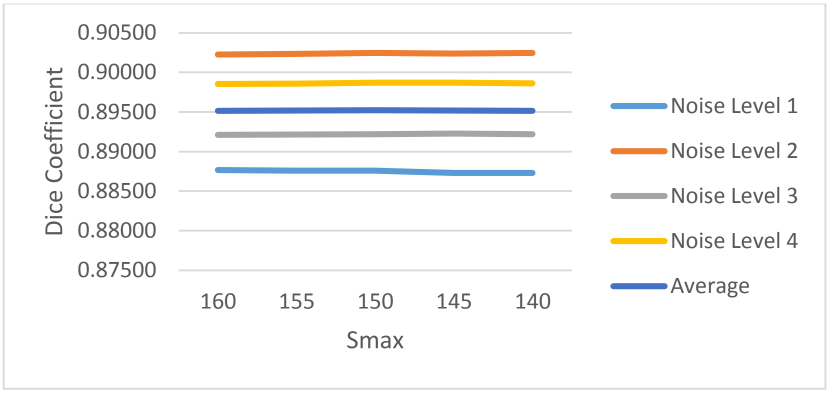
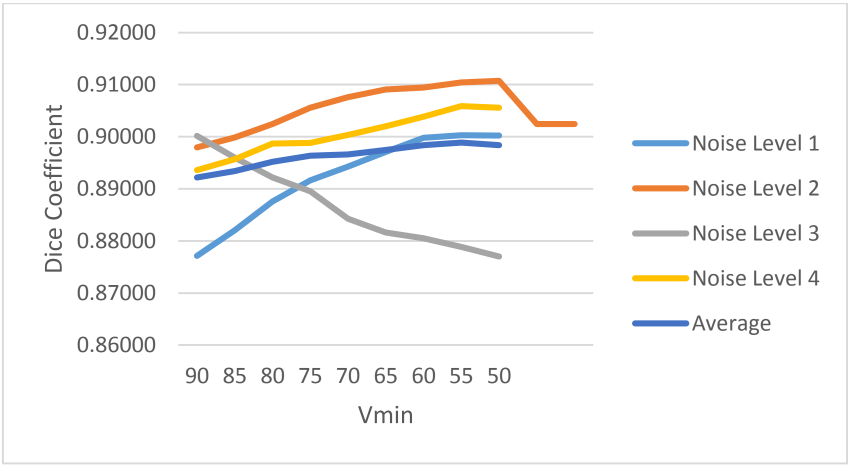


Figure 9 Training results on S values

In Figure 9, Smin is the lower threshold and it is trained from 20 to 50. Smax is the larger threshold and itis trained from 140 to 160. We can see that S value has a much larger range. The best value for Smin is 45 and the best value for Smax is 150.



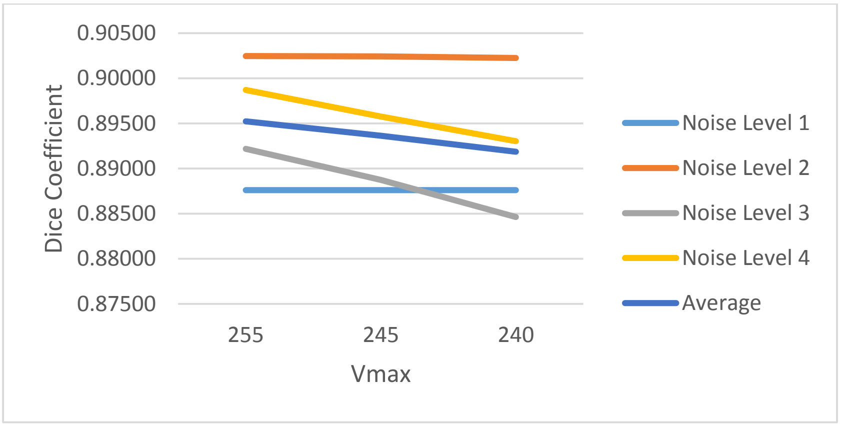


Figure 10 Training results on V values

In Figure 10, Vmin is the lower threshold and Vmax is the larger threshold. Vmin is trained from 50 to 90 and Vmax is trained from 240 to 255.We can see that V has the largest range compared with H and S values. The best value for Vmin is 55 and the best value for Vmax is 255.

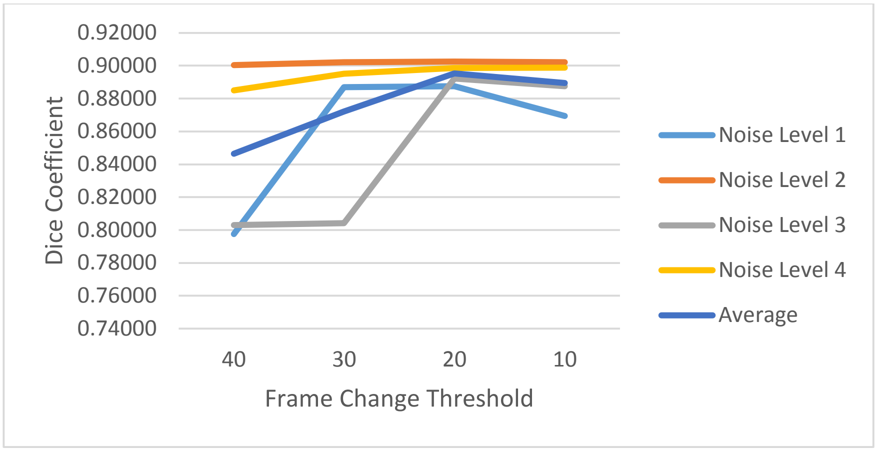


Figure 11 Training results on frame change threshold

For the frame change threshold value, shown in Figure 11, we can see that the best threshold is relatively small. The frame change threshold is trained from 10 to 40. The best threshold value is 20.

Finally, after the training process, the best set of parameters selected is H= [0, 20], S=[45,150], V=[55,255] and frame change threshold value is 20. The Dice coefficient average of the training sets with the original parameters is 89.52%. Using the optimal parameters, the Dice coefficient average improves from 89.52% to 90.85%. The image results are shown in Figure 12 and Figure 13.

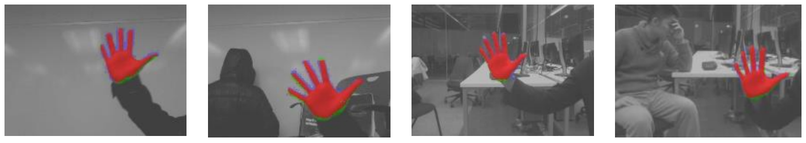


Figure 12 A sample frame of segmentation results on training sets with original parameters (noise level 1 to 4 from left to right)

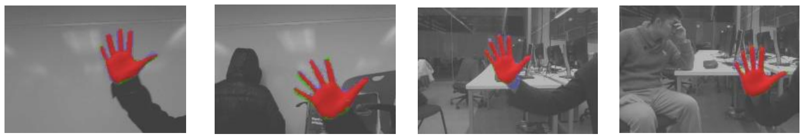


Figure 13 A sample frame of segmentation results on training sets with optimized parameters (noise level 1 to 4 from left to right)

Using this set of parameters, we evaluated the performance of the method on the testing sets. The result is shown in the table 1. The averaged accuracy (Dice Coefficient) is 87.75%. Figure 14 is the comparison image results using vrdiff in VisionX. Blue area is the false negative compared with the ground truth, green area is the false positive and the red area is the overlap.

Parameters	Noise Level 1	Noise Level 2	Noise Level 3	Noise Level 4	Average
H:0-20 **S:45-150** **V:55-255** **Threshold:20**	**0.88895**	**0.89576**	**0.86669**	**0.85853**	**0.87748**

Table 1 Results of Optimized segmentation method on testing sets

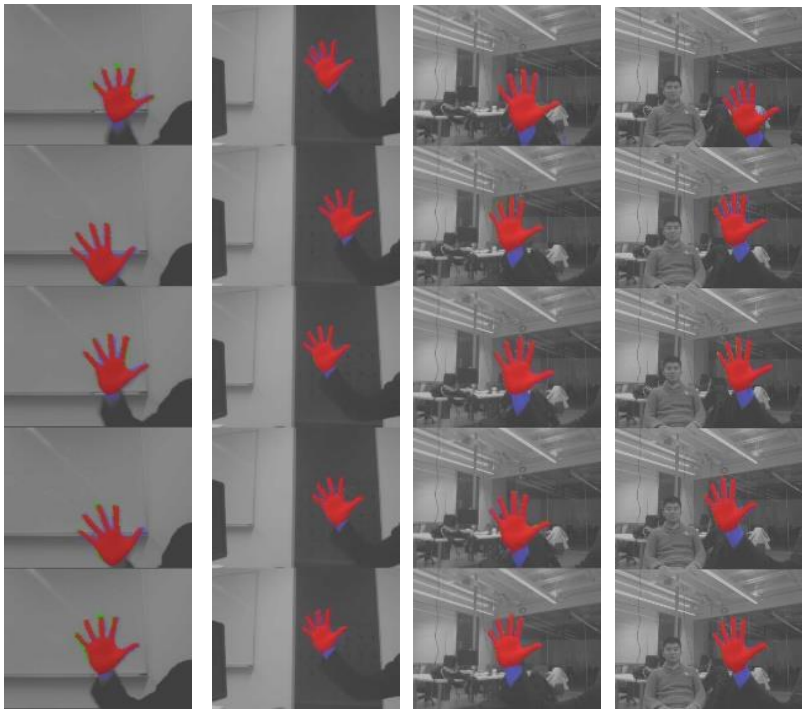


Figure 14 Final image comparison results using vrdiff for testing sets with optimized parameters

(noise level 1 to 4 from left to right)

4. CONCLUSION

In this project, we have implemented a hand segmentation method using multi-frame images. After careful research and studying, we successfully implemented the proposed method on training examples on various noise backgrounds under the assumption that with static background with certain illumination condition and moderate hand movement and no similar color objects around the hand. We then adjusted the parameters based on training results. Using the optimal parameters obtained from training process, the final result on training examples can be as high as 90.85% and for testing example it can be as high as 87.75%(Dice Coefficient). It turns out the algorithm works well and consistently under the similar condition as the training sets.

The HSV and change threshold values are optimal for training sets or similar condition. However, since the hand search algorithm (region of interest) increases the search region by one pixel all around every time, it may include some particles that are not part of the hand. i.e. any noise close to the hand will interfere the hand segmentation. Therefore future work could be optimizing the hand search algorithm or replace it with a different filter technique that would generate a better region of interest without considering where and what type of the noise is. Another thing can also be considered is to do more processing in the region of interest to filter other noise in the region of interest (shown in Figure 15). This might be done by applying the region of interest on the updated backprojection image rather than the thresholded image since backprojection gives a more accurate hand region (color similarity information). A new algorithm can be applied on the updated backprojection image to find the hand contour to thoroughly eliminate the noise behind the hand.

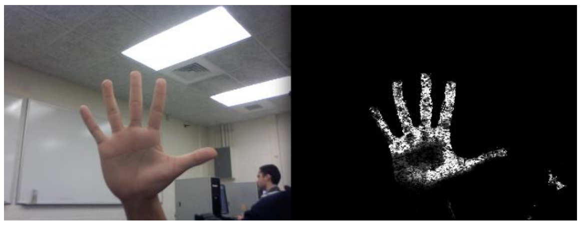


Figure 15 Suggested future work using backprojection

REFERENCES

[1] Kook-Yeol Yoo, “Robust Hand Segmentation and Tracking to Illumination Variation”, 2014 IEEE International Conference on Consumer Electronics (ICCE)

[2]Archana S. Ghotkar and Gajanan K. Kharate, “Hand Segmentation Techniques to Hand Gesture Recognition for Natural Human Computer Interaction”, International Journal of Human Computer Interaction (IJHCI), Volume (3) : Issue (1) : 2012

[3]Xintao Li, Can Tang, Chun Gong, Sheng Cheng and Jianwei Zhang, “Hand Segmentation Based on Skin Tone and Motion Detection with Complex Backgrounds,” Chinese Intelligent Automation Conference, Springer Berlin Heidelberg, Vol. 256, pp. 105~111, Jan. 2013.

[4]F. Chen, C. Fu, and C. Huang, “Hand Gesture Recognition Using a Real-Time Tracking Method and Hidden Markov Models,” Image and Video Computing, vol. 21, no. 8, pp. 745-758, Aug. 2003.

APPENDIX

Evaluation results on training examples

	H(max)	S(min)	S(max)	V(min)	V(max)	hangeThresho	Noise1	Noise2	Noise3	Noise4	avg
0	20	30	150	80	255	20	0.8876	0.90244	0.89218	0.89868	0.89522
5	20	30	150	80	255	20	0.86515	0.82108	0.89217	0.89793	0.86908
10	20	30	150	80	255	20	0.31122	0.19307	0.82823	0.84498	0.54438
0	**30**	30	150	80	255	20	0.88112	0.90199	0.88017	0.89193	0.8888
0	**25**	30	150	80	255	20	0.88221	0.90199	0.88413	0.894	0.89058
0	**15**	30	150	80	255	20	0.89521	0.90429	0.86781	0.86365	0.88274
0	**10**	30	150	80	255	20	0.82748	0.88343	0	0	0.42773
0	20	**40**	150	80	255	20	0.8963	0.91228	0.89267	0.90072	0.90049
0	20	**35**	150	80	255	20	0.89221	0.90814	0.89256	0.89935	0.89806
0	20	**25**	155	80	255	20	0.88343	0.89764	0.89155	0.89614	0.89219
0	20	**20**	150	80	255	20	0.87677	0.8918	0.89185	0.8956	0.889
0	20	30	**160**	80	255	20	0.88765	0.90227	0.89211	0.89856	0.89515
0	20	30	**155**	80	255	20	0.8876	0.90235	0.89216	0.89857	0.89517
0	20	30	**145**	80	255	20	0.88731	0.90239	0.89228	0.89869	0.89517
0	20	30	**140**	80	255	20	0.88732	0.90245	0.89218	0.89863	0.89514
0	20	30	150	**90**	255	20	0.8771	0.89795	0.90017	0.89362	0.89221
0	20	30	150	**85**	255	20	0.88206	0.89985	0.89597	0.89568	0.89339
0	20	30	150	**75**	255	20	0.89161	0.90558	0.88951	0.89879	0.89637
0	20	30	150	**70**	255	20	0.89428	0.90759	0.88423	0.90035	0.89661
0	20	30	150	80	**245**	20	0.8876	0.90244	0.88874	0.89574	0.89363
0	20	30	150	80	**240**	20	0.8876	0.90224	0.88462	0.89301	0.89187
0	20	30	150	80	255	**40**	0.79762	0.90031	0.80307	0.88504	0.84651
0	20	30	150	80	255	**30**	0.88699	0.90212	0.80414	0.89512	0.87209
0	20	30	150	80	255	**10**	0.86939	0.90216	0.88749	0.89883	0.88947
0	20	**45**	150	80	255	20	0.9006	0.91157	0.89417	0.90036	0.90168
0	20	30	150	**65**	255	20	0.8971	0.90907	0.88163	0.90199	0.89745
0	20	**50**	150	80	255	20	0.90468	0.90986	0.89118	0.89833	0.90102
0	20	30	150	**60**	255	20	0.89983	0.90942	0.88054	0.90384	0.89841
0	20	30	150	**55**	255	20	0.90029	0.91046	0.87885	0.90586	0.89887
0	20	30	150	**50**	255	20	0.90026	0.91072	0.87702	0.90557	0.89839

Program documentation

application.cpp

NAME

The core program that provides user input interface for mode selection and contains computer vision algorithm that produces segmentation output from the input resource defined by the user. This program controls the entire run procedures from the start to the end.

DESCRIPTION

testPara.txt is used to store prefered parameter sets that user want to use to run on the video sets if static mode is selected regardless training or testing. It must be store at /Users/BboyKellen/Documents/Xcode/headSeg/handSeg/handSeg/testPara.txt

At user level, this program asks user for inputs to determine the running mode.

<u>Static Mode:</u>

If train mode selected by the user, the following file must be presented in the directory to ensure correct functionality

/Users/BboyKellen/Documents/Xcode/headSeg/handSeg/handSeg/visionTrain.sh

input video source (train sets)with name of "noise1.mpeg","noise2.mpeg","noise3.mpeg","noise4.mpeg" should be stored in directory of /Users/BboyKellen/Documents/Xcode/headSeg/handSeg/

If test mode selected by the user, the following file must be presented in the directory to ensure correct functionality

/Users/BboyKellen/Documents/Xcode/headSeg/handSeg/handSeg/visionTest.sh

input video source (test sets) with name of "noiseTest1.mpeg", "noiseTest2.mpeg", "noiseTest3.mpeg", "noiseTest4.mpeg" should be stored in directory of /Users/BboyKellen/Documents/Xcode/headSeg/handSeg/The segmentation output videos will be stored in/Users/BboyKellen/Documents/Xcode/headSeg/handSeg/handSeg/with the video name of “output_” plus the input name.

<u>Real-time Mode:</u>

A active camera should be presented as part of the hardware.

AUTHOR

Xi He

visionTest.sh

NAME

The intermediate script program that is used to send test sets segmentation output videos to the ECE server account (jw859).

DESCRIPTION

If test mode selected by the user, this file must be presented in the directory to ensure correct functionality

/Users/BboyKellen/Documents/Xcode/headSeg/handSeg/handSeg/visionTest.sh

At each upload, test sets segmentation output videos with name of "output_noiseTest1.mpeg", "output_noiseTest2.mpeg", "output_noiseTest3.mpeg", "output_noiseTest4.mpeg" should be automatically stored by application.cpp in directory of /Users/BboyKellen/Documents/Xcode/headSeg/handSeg/handSeg/

The script will upload these videos to the ECE server account (jw859) to the directory of "/home/student/jw859"

AUTHOR

Xi He

visionTrain.sh

NAME

The intermediate script program that is used to send train sets segmentation output videos to the ECE server account (jw859).

DESCRIPTION

If train mode selected by the user, this file must be presented in the directory to ensure correct functionality

/Users/BboyKellen/Documents/Xcode/headSeg/handSeg/handSeg/visionTrain.sh

At each upload, train sets segmentation output videos with name of "output_noise1.mpeg", "output_noise2.mpeg", "output_noise3.mpeg", "output_noise4.mpeg" should be automatically stored by application.cpp in directory of /Users/BboyKellen/Documents/Xcode/headSeg/handSeg/handSeg/

The script will upload these videos to the ECE server account (jw859) to the directory of "/home/student/jw859"

AUTHOR

Xi He

EvaluationPara.sh

NAME

Produce the Dice Coefficient after comparing segmentation of training examples with ground truth.

DESCRIPTION

Given the segmentation files of training examples in home folder and the predefined boundary file, this script will produce the comparing results of the segmentation with ground truth. The segmentation files must be stored in home folder whose name is NetID. The files must be named as output_noise1.mpeg, output_noise2.mpeg, output_noise3.mpeg, output_noise4.mpeg

Predefined boundary files must be named as InzN1.vx with bookmark InzN1.1.vxa

InzN2.vx with bookmark InzN2.1.vxa, InzN3.vx with bookmark InzN3.1.vxa,

InzN4.vx with bookmark InzN4.1.vxa

Running this script will produce a output.txt which contains the Dice Coefficient for the training segmentation results in each line.

AUTHOR

Jilong Wu

EvaluationTest.sh

NAME

Produce the Dice Coefficient after comparing segmentation of testing examples with ground truth.

DESCRIPTION

Given the segmentation files of testing examples in home folder and the predefined boundary file, this script will produce the comparing results of the segmentation with ground truth. The segmentation files must be stored in home folder whose name is NetID. The files must be named as output_noiseTest1.mpeg, output_noiseTest2.mpeg, output_noiseTest3.mpeg, output_noiseTest4.mpeg

Predefined boundary files must be named as InzT1.vx with bookmark InzT1.1.vxa

InzT2.vx with bookmark InzT2.1.vxa, InzT3.vx with bookmark InzT3.1.vxa,

InzT4.vx with bookmark InzT4.1.vxa

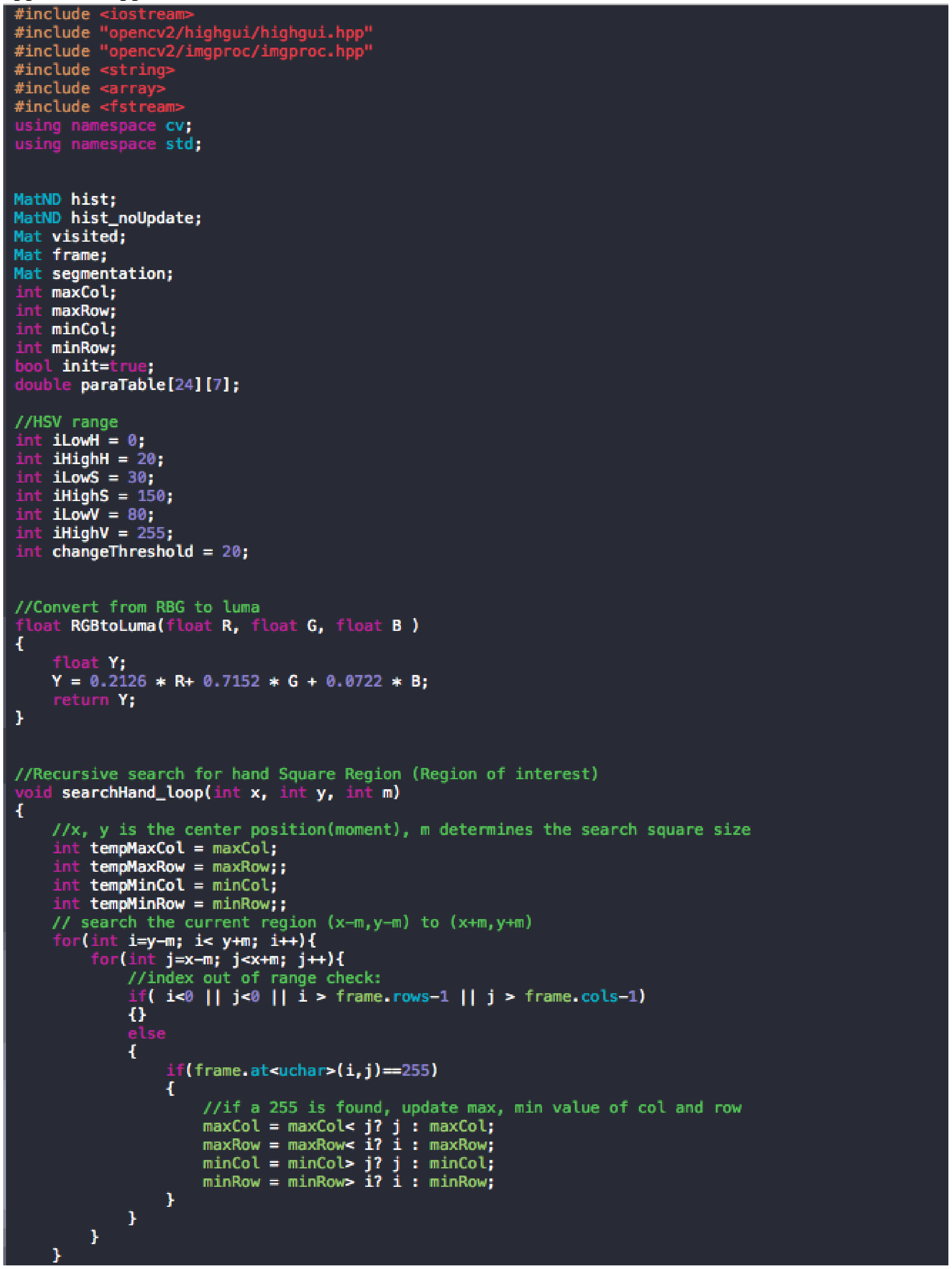
Running this script will produce a outputT.txt which contains the Dice Coefficient for the testing segmentation results in each line.

AUTHOR

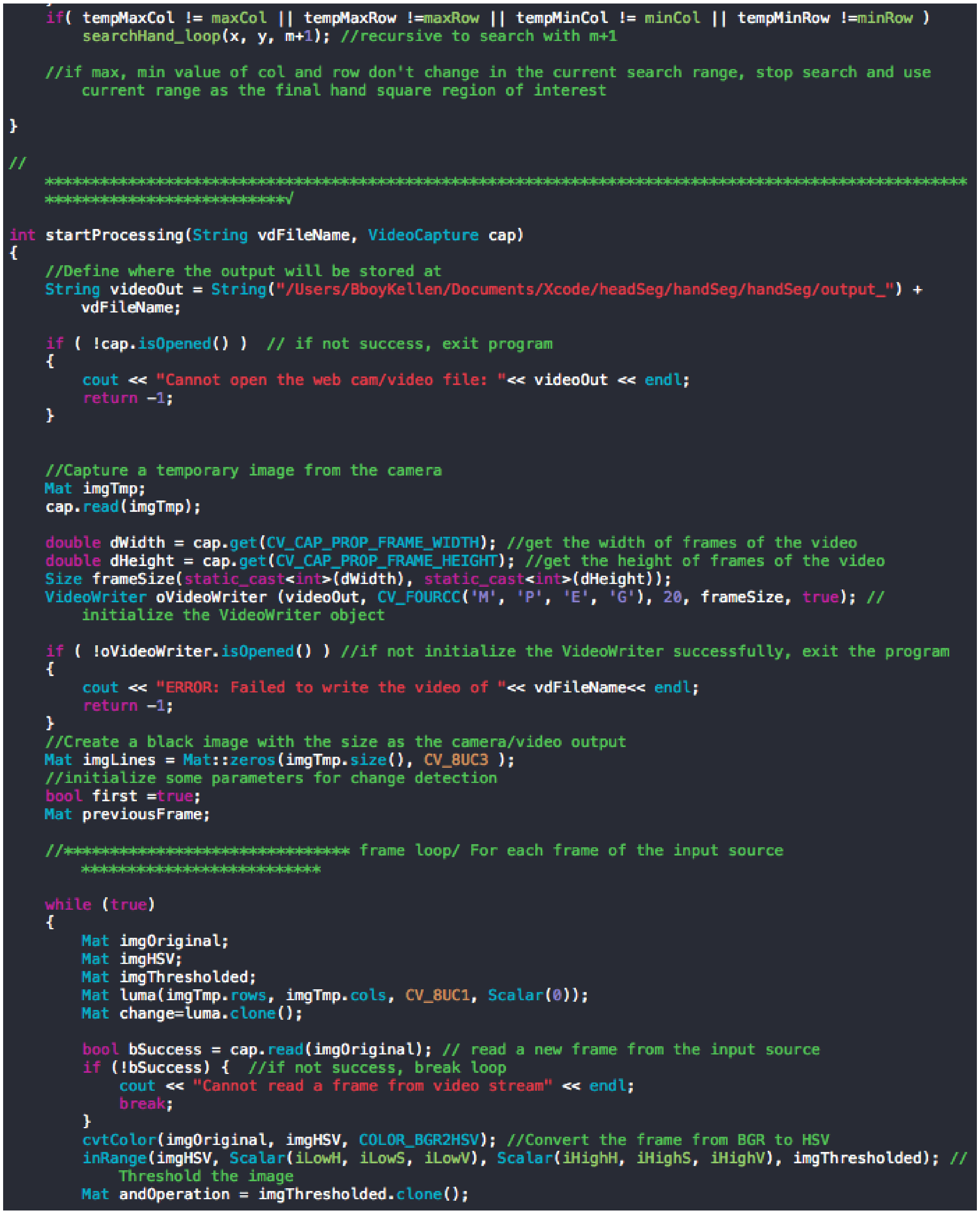
Jilong Wu

List of programs:

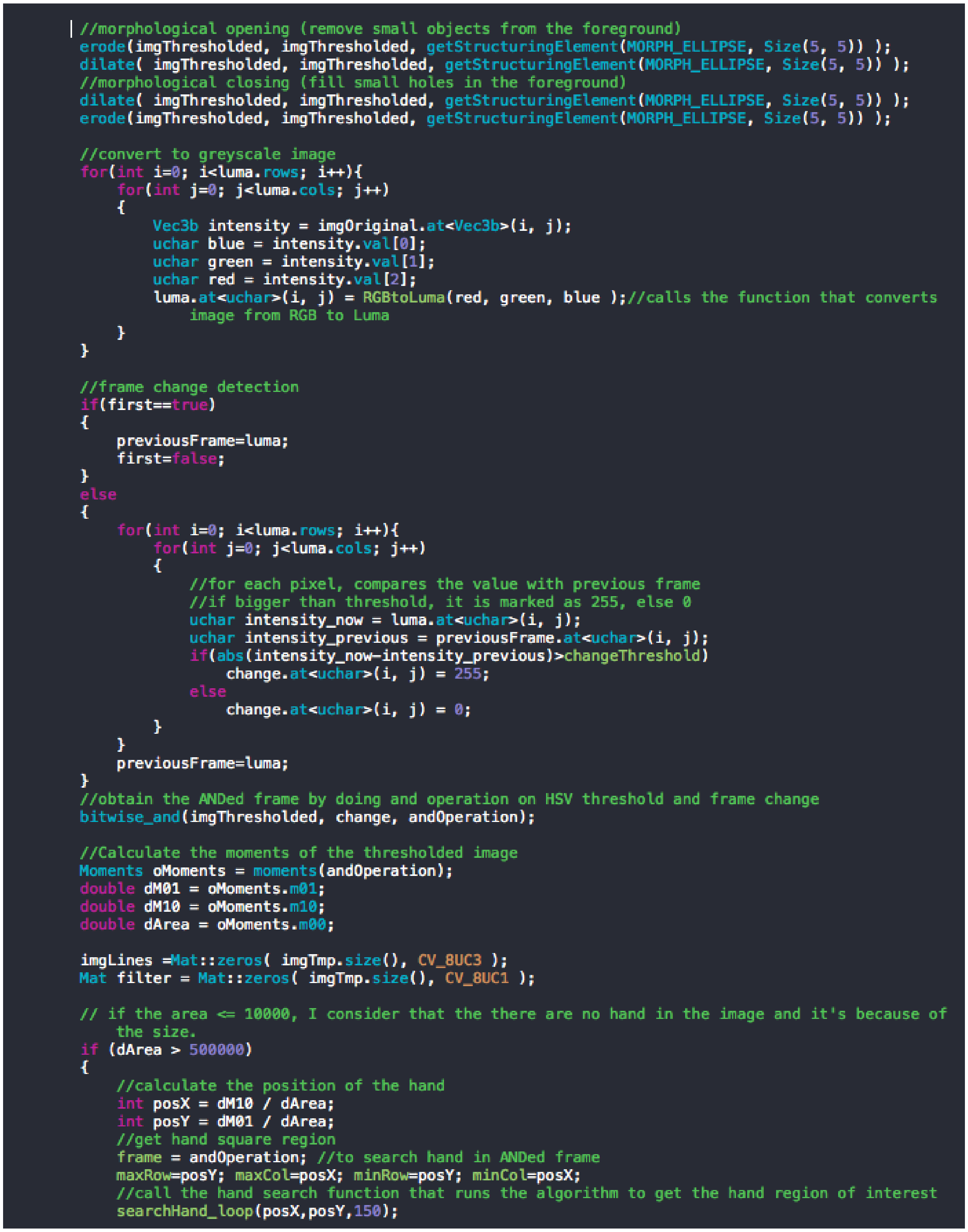
application.cpp



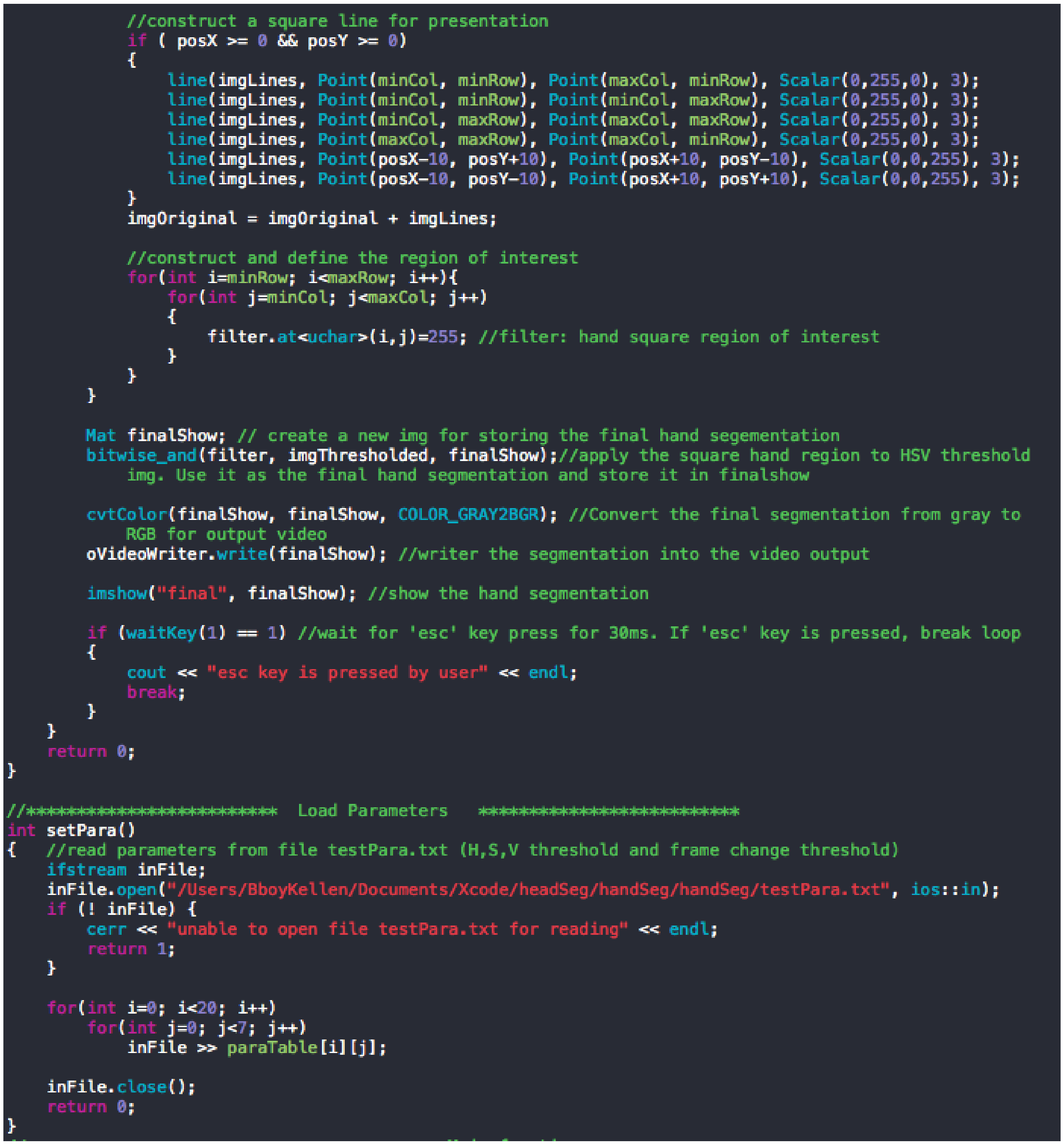
```
#include <iostream>
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include <string>
#include <array>
#include <fstream>
using namespace cv;
using namespace std;


MatND hist;
MatND hist_noUpdate;
Mat visited;
Mat frame;
Mat segmentation;
int maxCol;
int maxRow;
int minCol;
int minRow;
bool init=true;
double paraTable[24][7];

//HSV range
int iLowH = 0;
int iHighH = 20;
int iLowS = 30;
int iHighS = 150;
int iLowV = 80;
int iHighV = 255;
int changeThreshold = 20;


//Convert from RBG to luma
float RGBtoLuma(float R, float G, float B )
{
    float Y;
    Y = 0.2126 * R+ 0.7152 * G + 0.0722 * B;
    return Y;
}


//Recursive search for hand Square Region (Region of interest)
void searchHand_loop(int x, int y, int m)
{
    //x, y is the center position(moment), m determines the search square size
    int tempMaxCol = maxCol;
    int tempMaxRow = maxRow;;
    int tempMinCol = minCol;
    int tempMinRow = minRow;;
    // search the current region (x-m,y-m) to (x+m,y+m)
    for(int i=y-m; i< y+m; i++){
        for(int j=x-m; j<x+m; j++){
            //index out of range check:
            if( i<0 || j<0 || i > frame.rows-1 || j > frame.cols-1)
            {}
            else
            {
                if(frame.at<uchar>(i,j)==255)
                {
                    //if a 255 is found, update max, min value of col and row
                    maxCol = maxCol< j? j : maxCol;
                    maxRow = maxRow< i? i : maxRow;
                    minCol = minCol> j? j : minCol;
                    minRow = minRow> i? i : minRow;
                }
            }
        }
    }
```



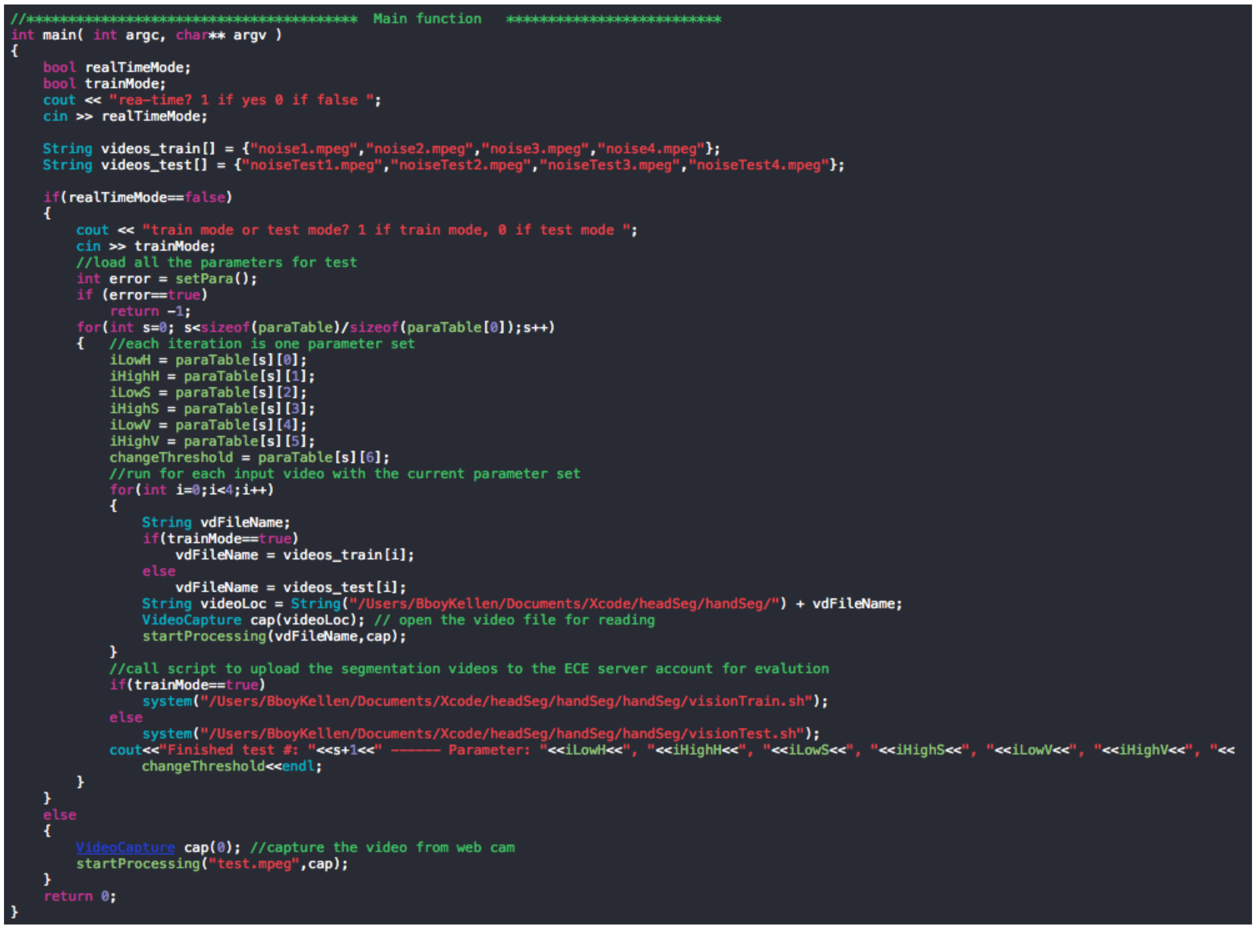
```
    if( tempMaxCol != maxCol || tempMaxRow !=maxRow || tempMinCol != minCol || tempMinRow !=minRow )
        searchHand_loop(x, y, m+1); //recursive to search with m+1

    //if max, min value of col and row don't change in the current search range, stop search and use
        current range as the final hand square region of interest

}

//
    *************************************************************************************************
    ***************************/

int startProcessing(String vdFileName, VideoCapture cap)
{
    //Define where the output will be stored at
    String videoOut = String("/Users/BboyKellen/Documents/Xcode/headSeg/handSeg/handSeg/output_") +
        vdFileName;

    if ( !cap.isOpened() )  // if not success, exit program
    {
        cout << "Cannot open the web cam/video file: "<< videoOut << endl;
        return -1;
    }


    //Capture a temporary image from the camera
    Mat imgTmp;
    cap.read(imgTmp);

    double dWidth = cap.get(CV_CAP_PROP_FRAME_WIDTH); //get the width of frames of the video
    double dHeight = cap.get(CV_CAP_PROP_FRAME_HEIGHT); //get the height of frames of the video
    Size frameSize(static_cast<int>(dWidth), static_cast<int>(dHeight));
    VideoWriter oVideoWriter (videoOut, CV_FOURCC('M', 'P', 'E', 'G'), 20, frameSize, true); //
        initialize the VideoWriter object

    if ( !oVideoWriter.isOpened() ) //if not initialize the VideoWriter successfully, exit the program
    {
        cout << "ERROR: Failed to write the video of "<< vdFileName<< endl;
        return -1;
    }
    //Create a black image with the size as the camera/video output
    Mat imgLines = Mat::zeros(imgTmp.size(), CV_8UC3 );
    //initialize some parameters for change detection
    bool first =true;
    Mat previousFrame;

    //******************************** frame loop/ For each frame of the input source
        ***************************

    while (true)
    {
        Mat imgOriginal;
        Mat imgHSV;
        Mat imgThresholded;
        Mat luma(imgTmp.rows, imgTmp.cols, CV_8UC1, Scalar(0));
        Mat change=luma.clone();

        bool bSuccess = cap.read(imgOriginal); // read a new frame from the input source
        if (!bSuccess) {  //if not success, break loop
            cout << "Cannot read a frame from video stream" << endl;
            break;
        }
        cvtColor(imgOriginal, imgHSV, COLOR_BGR2HSV); //Convert the frame from BGR to HSV
        inRange(imgHSV, Scalar(iLowH, iLowS, iLowV), Scalar(iHighH, iHighS, iHighV), imgThresholded); //
            Threshold the image
        Mat andOperation = imgThresholded.clone();
```



```
//morphological opening (remove small objects from the foreground)
erode(imgThresholded, imgThresholded, getStructuringElement(MORPH_ELLIPSE, Size(5, 5)) );
dilate( imgThresholded, imgThresholded, getStructuringElement(MORPH_ELLIPSE, Size(5, 5)) );
//morphological closing (fill small holes in the foreground)
dilate( imgThresholded, imgThresholded, getStructuringElement(MORPH_ELLIPSE, Size(5, 5)) );
erode(imgThresholded, imgThresholded, getStructuringElement(MORPH_ELLIPSE, Size(5, 5)) );

//convert to greyscale image
for(int i=0; i<luma.rows; i++){
    for(int j=0; j<luma.cols; j++)
    {
        Vec3b intensity = imgOriginal.at<Vec3b>(i, j);
        uchar blue = intensity.val[0];
        uchar green = intensity.val[1];
        uchar red = intensity.val[2];
        luma.at<uchar>(i, j) = RGBtoLuma(red, green, blue );//calls the function that converts
            image from RGB to Luma
    }
}

//frame change detection
if(first==true)
{
    previousFrame=luma;
    first=false;
}
else
{
    for(int i=0; i<luma.rows; i++){
        for(int j=0; j<luma.cols; j++)
        {
            //for each pixel, compares the value with previous frame
            //if bigger than threshold, it is marked as 255, else 0
            uchar intensity_now = luma.at<uchar>(i, j);
            uchar intensity_previous = previousFrame.at<uchar>(i, j);
            if(abs(intensity_now-intensity_previous)>changeThreshold)
                change.at<uchar>(i, j) = 255;
            else
                change.at<uchar>(i, j) = 0;
        }
    }
    previousFrame=luma;
}
//obtain the ANDed frame by doing and operation on HSV threshold and frame change
bitwise_and(imgThresholded, change, andOperation);

//Calculate the moments of the thresholded image
Moments oMoments = moments(andOperation);
double dM01 = oMoments.m01;
double dM10 = oMoments.m10;
double dArea = oMoments.m00;

imgLines =Mat::zeros( imgTmp.size(), CV_8UC3 );
Mat filter = Mat::zeros( imgTmp.size(), CV_8UC1 );

// if the area <= 10000, I consider that the there are no hand in the image and it's because of
    the size.
if (dArea > 500000)
{
    //calculate the position of the hand
    int posX = dM10 / dArea;
    int posY = dM01 / dArea;
    //get hand square region
    frame = andOperation; //to search hand in ANDed frame
    maxRow=posY; maxCol=posX; minRow=posY; minCol=posX;
    //call the hand search function that runs the algorithm to get the hand region of interest
    searchHand_loop(posX,posY,150);
```

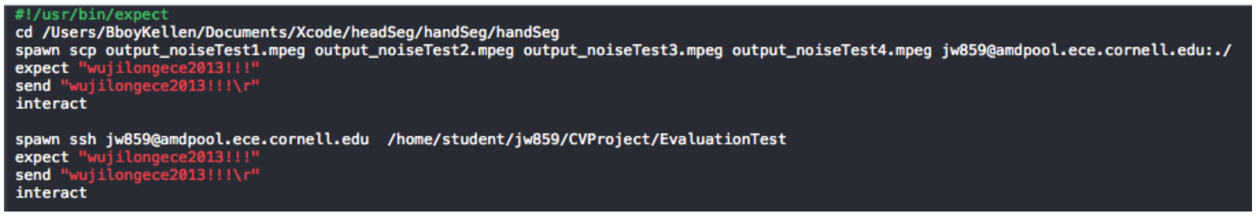


```
            //construct a square line for presentation
            if ( posX >= 0 && posY >= 0)
            {
                line(imgLines, Point(minCol, minRow), Point(maxCol, minRow), Scalar(0,255,0), 3);
                line(imgLines, Point(minCol, minRow), Point(minCol, maxRow), Scalar(0,255,0), 3);
                line(imgLines, Point(minCol, maxRow), Point(maxCol, maxRow), Scalar(0,255,0), 3);
                line(imgLines, Point(maxCol, maxRow), Point(maxCol, minRow), Scalar(0,255,0), 3);
                line(imgLines, Point(posX-10, posY+10), Point(posX+10, posY-10), Scalar(0,0,255), 3);
                line(imgLines, Point(posX-10, posY-10), Point(posX+10, posY+10), Scalar(0,0,255), 3);
            }
            imgOriginal = imgOriginal + imgLines;

            //construct and define the region of interest
            for(int i=minRow; i<maxRow; i++){
                for(int j=minCol; j<maxCol; j++)
                {
                    filter.at<uchar>(i,j)=255; //filter: hand square region of interest
                }
            }
        }

        Mat finalShow; // create a new img for storing the final hand segementation
        bitwise_and(filter, imgThresholded, finalShow);//apply the square hand region to HSV threshold
            img. Use it as the final hand segmentation and store it in finalshow

        cvtColor(finalShow, finalShow, COLOR_GRAY2BGR); //Convert the final segmentation from gray to
            RGB for output video
        oVideoWriter.write(finalShow); //writer the segmentation into the video output

        imshow("final", finalShow); //show the hand segmentation

        if (waitKey(1) == 1) //wait for 'esc' key press for 30ms. If 'esc' key is pressed, break loop
        {
            cout << "esc key is pressed by user" << endl;
            break;
        }
    }
    return 0;
}

//*************************  Load Parameters   *************************
int setPara()
{   //read parameters from file testPara.txt (H,S,V threshold and frame change threshold)
    ifstream inFile;
    inFile.open("/Users/BboyKellen/Documents/Xcode/headSeg/handSeg/handSeg/testPara.txt", ios::in);
    if (! inFile) {
        cerr << "unable to open file testPara.txt for reading" << endl;
        return 1;
    }

    for(int i=0; i<20; i++)
        for(int j=0; j<7; j++)
            inFile >> paraTable[i][j];

    inFile.close();
    return 0;
}
```



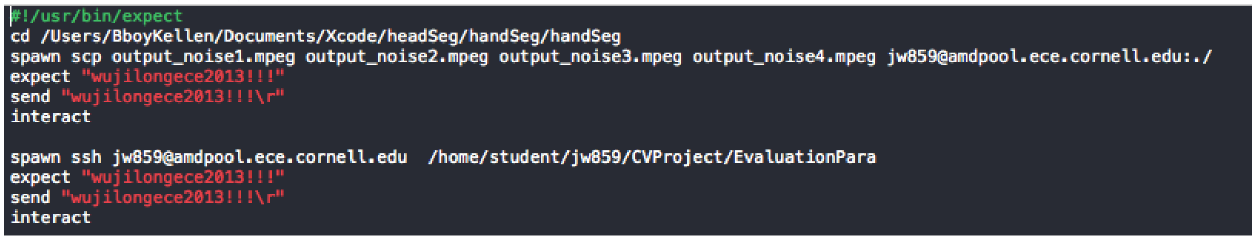
```
//*****************************************  Main function   ***************************
int main( int argc, char** argv )
{
    bool realTimeMode;
    bool trainMode;
    cout << "rea-time? 1 if yes 0 if false ";
    cin >> realTimeMode;

    String videos_train[] = {"noise1.mpeg","noise2.mpeg","noise3.mpeg","noise4.mpeg"};
    String videos_test[] = {"noiseTest1.mpeg","noiseTest2.mpeg","noiseTest3.mpeg","noiseTest4.mpeg"};

    if(realTimeMode==false)
    {
        cout << "train mode or test mode? 1 if train mode, 0 if test mode ";
        cin >> trainMode;
        //load all the parameters for test
        int error = setPara();
        if (error==true)
            return -1;
        for(int s=0; s<sizeof(paraTable)/sizeof(paraTable[0]);s++)
        {   //each iteration is one parameter set
            iLowH = paraTable[s][0];
            iHighH = paraTable[s][1];
            iLowS = paraTable[s][2];
            iHighS = paraTable[s][3];
            iLowV = paraTable[s][4];
            iHighV = paraTable[s][5];
            changeThreshold = paraTable[s][6];
            //run for each input video with the current parameter set
            for(int i=0;i<4;i++)
            {
                String vdFileName;
                if(trainMode==true)
                    vdFileName = videos_train[i];
                else
                    vdFileName = videos_test[i];
                String videoLoc = String("/Users/BboyKellen/Documents/Xcode/headSeg/handSeg/") + vdFileName;
                VideoCapture cap(videoLoc); // open the video file for reading
                startProcessing(vdFileName,cap);
            }
            //call script to upload the segmentation videos to the ECE server account for evalution
            if(trainMode==true)
                system("/Users/BboyKellen/Documents/Xcode/headSeg/handSeg/handSeg/visionTrain.sh");
            else
                system("/Users/BboyKellen/Documents/Xcode/headSeg/handSeg/handSeg/visionTest.sh");
            cout<<"Finished test #: "<<s+1<<" ------ Parameter: "<<iLowH<<", "<<iHighH<<", "<<iLowS<<", "<<iHighS<<", "<<iLowV<<", "<<iHighV<<", "<<
                changeThreshold<<endl;
        }
    }
    else
    {
        VideoCapture cap(0); //capture the video from web cam
        startProcessing("test.mpeg",cap);
    }
    return 0;
}
```

visionTest.sh



```
#!/usr/bin/expect
cd /Users/BboyKellen/Documents/Xcode/headSeg/handSeg/handSeg
spawn scp output_noiseTest1.mpeg output_noiseTest2.mpeg output_noiseTest3.mpeg output_noiseTest4.mpeg jw859@amdpool.ece.cornell.edu:./
expect "wujilongece2013!!!"
send "wujilongece2013!!!\r"
interact

spawn ssh jw859@amdpool.ece.cornell.edu  /home/student/jw859/CVProject/EvaluationTest
expect "wujilongece2013!!!"
send "wujilongece2013!!!\r"
interact
```

visionTrain.sh



```
#!/usr/bin/expect
cd /Users/BboyKellen/Documents/Xcode/headSeg/handSeg/handSeg
spawn scp output_noise1.mpeg output_noise2.mpeg output_noise3.mpeg output_noise4.mpeg jw859@amdpool.ece.cornell.edu:./
expect "wujilongece2013!!!"
send "wujilongece2013!!!\r"
interact

spawn ssh jw859@amdpool.ece.cornell.edu  /home/student/jw859/CVProject/EvaluationPara
expect "wujilongece2013!!!"
send "wujilongece2013!!!\r"
interact
```

EvaluationPara:

```
export PATH=$PATH:/home/student/jw859/install/stow-pkgs/x86_64-rhel5/bin:/home/student/jw859/install/stow-pkgs/noarch/bin:/research/brg/install/stow-pkgs/x86_64-rhel5/bin:/research/brg/install/stow-
```

```
pkgs/noarch/bin:/opt/alttools/bin:/opt/synopsys/F-
2011.06/bin:/usr/kerberos/bin:/usr/local/bin:/bin:/usr/bin:/classes/ece547/32/v4/bin:/cla
sses/ece547/bin:/opt/sonnet/14.54/bin:/opt/ads/ADS2011_10/bin:/opt/ansoft/hfss15.0/Linux:
/opt/ansys/ANSYS-
13.0SP2/v130/ansys/bin:/opt/cadence/ic/bin:/opt/cadence/ic/tools/dfII/bin:/opt/cadence/ic
/tools/bin:/opt/cadence/ius/tools.lnx86/bin:/opt/cadence/assura/bin:/opt/cadence/ccd/bin:
/opt/cadence/confrml/bin:/opt/cadence/cva/bin:/opt/cadence/eanl/bin:/opt/cadence/et/bin:/
opt/cadence/ets/bin:/opt/cadence/ext/bin:/opt/cadence/finale/bin:/opt/cadence/icoa/bin:/o
pt/cadence/ixe/bin:/opt/cadence/mmsim/bin:/opt/cadence/pacific/bin:/opt/cadence/pvs/bin:/
opt/cadence/rc/bin:/opt/cadence/ret/bin:/opt/cadence/sev/bin:/opt/cadence/soc/bin:/opt/ca
dence/spmn/bin:/opt/cadence/tsi/bin:/opt/cadence/xae/bin:/opt/cadence/amskit/tools.lnx86/
bin:/opt/cadence/cwspw/tools.lnx86/bin:/opt/cadence/icc/tools.lnx86/bin:/opt/cadence/ifv/
tools.lnx86/bin:/opt/cadence/isv/tools.lnx86/bin:/opt/cadence/ius/tools.lnx86/bin:/opt/ca
dence/kcl/tools.lnx86/bin:/opt/cadence/kmc/tools.lnx86/bin:/opt/cadence/kqv/tools.lnx86/b
in:/opt/cadence/lcu/tools.lnx86/bin:/opt/cadence/neocell/tools.lnx86/bin:/opt/cadence/neo
ckt/tools.lnx86/bin:/opt/cadence/psd/tools.lnx86/bin:/opt/cadence/rfkit/tools.lnx86/bin:/
opt/cadence/spb/tools.lnx86/bin:/opt/cadence/tda/tools.lnx86/bin:/opt/cadence/tvm/tools.l
nx86/bin:/opt/cadence/usim/tools.lnx86/bin:/opt/cadence/vce/tools.lnx86/bin:/opt/cadence/
vip/tools.lnx86/bin:/opt/cadence/vsde/tools.lnx86/bin:/opt/comsol/4.2/bin:/classes/brg/in
stall/noarch/bin:/opt/alttools/ece:/classes/ece314/bin:/classes/ece475/bin:/classes/ece54
7/32/v4/bin:/classes/ece547/bin:/opt/Magic/magic-
7.3.108/x86_64/bin:/opt/matlab/R2012a/bin:/opt/QuartusII/10.0SP1/bin:/opt/silvaco/2010-
00/bin/:/opt/synplcty/fpga_94/bin:/opt/synplcty/identify_30/bin:/opt/synplcty/system_desi
gner_202/bin
cd /home/student/jw859/CVProject
cp /home/student/jw859/output_noise1.mpeg
/home/student/jw859/CVProject/output_noise1.mpeg
cp /home/student/jw859/output_noise2.mpeg
/home/student/jw859/CVProject/output_noise2.mpeg
cp /home/student/jw859/output_noise3.mpeg
/home/student/jw859/CVProject/output_noise3.mpeg
cp /home/student/jw859/output_noise4.mpeg
/home/student/jw859/CVProject/output_noise4.mpeg

vmpegtovx -g f=1,30 if=output_noise1.mpeg of=noise1Out.vx
vclip f=6 if=noise1Out.vx of=6O.vx
vclip f=11 if=noise1Out.vx of=11O.vx
vclip f=16 if=noise1Out.vx of=16O.vx
vclip f=21 if=noise1Out.vx of=21O.vx
vclip f=26 if=noise1Out.vx of=26O.vx
vcat if=6O.vx 11O.vx 16O.vx 21O.vx 26O.vx of=Out.vx
vdim -c if=Out.vx of=Outz.vx
vrdiff if=InzN1.1.vxa bf=Outz.vx -cp of=compareN1_1.vx ig=InzN1.vx os=compareN1_1.txt



vmpegtovx -g f=1,30 if=output_noise2.mpeg of=noise1Out.vx
vclip f=6 if=noise1Out.vx of=6O.vx
vclip f=11 if=noise1Out.vx of=11O.vx
vclip f=16 if=noise1Out.vx of=16O.vx
vclip f=21 if=noise1Out.vx of=21O.vx
vclip f=26 if=noise1Out.vx of=26O.vx
vcat if=6O.vx 11O.vx 16O.vx 21O.vx 26O.vx of=Out.vx
vdim -c if=Out.vx of=Outz.vx
vrdiff if=InzN2.1.vxa bf=Outz.vx -cp of=compareN2_1.vx ig=InzN2.vx os=compareN2_1.txt
```

```
vmpegtovx -g f=1,30 if=output_noise3.mpeg of=noise1Out.vx
vclip f=6 if=noise1Out.vx of=6O.vx
vclip f=11 if=noise1Out.vx of=11O.vx
vclip f=16 if=noise1Out.vx of=16O.vx
vclip f=21 if=noise1Out.vx of=21O.vx
vclip f=26 if=noise1Out.vx of=26O.vx
vcat if=6O.vx 11O.vx 16O.vx 21O.vx 26O.vx of=Out.vx
vdim -c if=Out.vx of=Outz.vx
vrdiff if=InzN3.1.vxa bf=Outz.vx -cp of=compareN3_1.vx ig=InzN3.vx os=compareN3_1.txt




vmpegtovx -g f=1,30 if=output_noise4.mpeg of=noise1Out.vx
vclip f=6 if=noise1Out.vx of=6O.vx
vclip f=11 if=noise1Out.vx of=11O.vx
vclip f=16 if=noise1Out.vx of=16O.vx
vclip f=21 if=noise1Out.vx of=21O.vx
vclip f=26 if=noise1Out.vx of=26O.vx
vcat if=6O.vx 11O.vx 16O.vx 21O.vx 26O.vx of=Out.vx
vdim -c if=Out.vx of=Outz.vx
vrdiff if=InzN4.1.vxa bf=Outz.vx -cp of=compareN4_1.vx ig=InzN4.vx os=compareN4_1.txt


a=`tail -1 compareN1_1.txt | sed 's/dsc//g'`
b=`tail -1 compareN2_1.txt | sed 's/dsc//g'`
c=`tail -1 compareN3_1.txt | sed 's/dsc//g'`
d=`tail -1 compareN4_1.txt | sed 's/dsc//g'`
new=$a$b$c$d
echo $new >> output.txt
```

EvaluationTest

```
export PATH=$PATH:/home/student/jw859/install/stow-pkgs/x86_64-
rhel5/bin:/home/student/jw859/install/stow-pkgs/noarch/bin:/research/brg/install/stow-
pkgs/x86_64-rhel5/bin:/research/brg/install/stow-
pkgs/noarch/bin:/opt/alttools/bin:/opt/synopsys/F-
2011.06/bin:/usr/kerberos/bin:/usr/local/bin:/bin:/usr/bin:/classes/ece547/32/v4/bin:/cla
sses/ece547/bin:/opt/sonnet/14.54/bin:/opt/ads/ADS2011_10/bin:/opt/ansoft/hfss15.0/Linux:
/opt/ansys/ANSYS-
13.0SP2/v130/ansys/bin:/opt/cadence/ic/bin:/opt/cadence/ic/tools/dfII/bin:/opt/cadence/ic
/tools/bin:/opt/cadence/ius/tools.lnx86/bin:/opt/cadence/assura/bin:/opt/cadence/ccd/bin:
/opt/cadence/confrml/bin:/opt/cadence/cva/bin:/opt/cadence/eanl/bin:/opt/cadence/et/bin:/
opt/cadence/ets/bin:/opt/cadence/ext/bin:/opt/cadence/finale/bin:/opt/cadence/icoa/bin:/o
pt/cadence/ixe/bin:/opt/cadence/mmsim/bin:/opt/cadence/pacific/bin:/opt/cadence/pvs/bin:/
opt/cadence/rc/bin:/opt/cadence/ret/bin:/opt/cadence/sev/bin:/opt/cadence/soc/bin:/opt/ca
dence/spmn/bin:/opt/cadence/tsi/bin:/opt/cadence/xae/bin:/opt/cadence/amskit/tools.lnx86/
bin:/opt/cadence/cwspw/tools.lnx86/bin:/opt/cadence/icc/tools.lnx86/bin:/opt/cadence/ifv/
tools.lnx86/bin:/opt/cadence/isv/tools.lnx86/bin:/opt/cadence/ius/tools.lnx86/bin:/opt/ca
dence/kcl/tools.lnx86/bin:/opt/cadence/kmc/tools.lnx86/bin:/opt/cadence/kqv/tools.lnx86/b
in:/opt/cadence/lcu/tools.lnx86/bin:/opt/cadence/neocell/tools.lnx86/bin:/opt/cadence/neo
ckt/tools.lnx86/bin:/opt/cadence/psd/tools.lnx86/bin:/opt/cadence/rfkit/tools.lnx86/bin:/
opt/cadence/spb/tools.lnx86/bin:/opt/cadence/tda/tools.lnx86/bin:/opt/cadence/tvm/tools.l
nx86/bin:/opt/cadence/usim/tools.lnx86/bin:/opt/cadence/vce/tools.lnx86/bin:/opt/cadence/
vip/tools.lnx86/bin:/opt/cadence/vsde/tools.lnx86/bin:/opt/comsol/4.2/bin:/classes/brg/in
stall/noarch/bin:/opt/alttools/ece:/classes/ece314/bin:/classes/ece475/bin:/classes/ece54
7/32/v4/bin:/classes/ece547/bin:/opt/Magic/magic-
7.3.108/x86_64/bin:/opt/matlab/R2012a/bin:/opt/QuartusII/10.0SP1/bin:/opt/silvaco/2010-
```

```
00/bin/:/opt/synplcty/fpga_94/bin:/opt/synplcty/identify_30/bin:/opt/synplcty/system_desi
gner_202/bin
cd /home/student/jw859/CVProject
cp /home/student/jw859/output_noiseTest1.mpeg
/home/student/jw859/CVProject/output_noiseTest1.mpeg
cp /home/student/jw859/output_noiseTest2.mpeg
/home/student/jw859/CVProject/output_noiseTest2.mpeg
cp /home/student/jw859/output_noiseTest3.mpeg
/home/student/jw859/CVProject/output_noiseTest3.mpeg
cp /home/student/jw859/output_noiseTest4.mpeg
/home/student/jw859/CVProject/output_noiseTest4.mpeg

vmpegtovx -g f=1,30 if=output_noiseTest1.mpeg of=noise1Out.vx
vclip f=6 if=noise1Out.vx of=6O.vx
vclip f=11 if=noise1Out.vx of=11O.vx
vclip f=16 if=noise1Out.vx of=16O.vx
vclip f=21 if=noise1Out.vx of=21O.vx
vclip f=26 if=noise1Out.vx of=26O.vx
vcat if=6O.vx 11O.vx 16O.vx 21O.vx 26O.vx of=Out.vx
vdim -c if=Out.vx of=Outz.vx
vrdiff if=InzT1.1.vxa bf=Outz.vx -cp of=compareT1_1.vx ig=InzT1.vx os=compareT1_1.txt



vmpegtovx -g f=1,30 if=output_noiseTest2.mpeg of=noise1Out.vx
vclip f=6 if=noise1Out.vx of=6O.vx
vclip f=11 if=noise1Out.vx of=11O.vx
vclip f=16 if=noise1Out.vx of=16O.vx
vclip f=21 if=noise1Out.vx of=21O.vx
vclip f=26 if=noise1Out.vx of=26O.vx
vcat if=6O.vx 11O.vx 16O.vx 21O.vx 26O.vx of=Out.vx
vdim -c if=Out.vx of=Outz.vx
vrdiff if=InzT2.1.vxa bf=Outz.vx -cp of=compareT2_1.vx ig=InzT2.vx os=compareT2_1.txt


vmpegtovx -g f=1,30 if=output_noiseTest3.mpeg of=noise1Out.vx
vclip f=6 if=noise1Out.vx of=6O.vx
vclip f=11 if=noise1Out.vx of=11O.vx
vclip f=16 if=noise1Out.vx of=16O.vx
vclip f=21 if=noise1Out.vx of=21O.vx
vclip f=26 if=noise1Out.vx of=26O.vx
vcat if=6O.vx 11O.vx 16O.vx 21O.vx 26O.vx of=Out.vx
vdim -c if=Out.vx of=Outz.vx
vrdiff if=InzT3.1.vxa bf=Outz.vx -cp of=compareT3_1.vx ig=InzT3.vx os=compareT3_1.txt



vmpegtovx -g f=1,30 if=output_noiseTest4.mpeg of=noise1Out.vx
vclip f=6 if=noise1Out.vx of=6O.vx
vclip f=11 if=noise1Out.vx of=11O.vx
vclip f=16 if=noise1Out.vx of=16O.vx
vclip f=21 if=noise1Out.vx of=21O.vx
vclip f=26 if=noise1Out.vx of=26O.vx
vcat if=6O.vx 11O.vx 16O.vx 21O.vx 26O.vx of=Out.vx
vdim -c if=Out.vx of=Outz.vx
vrdiff if=InzT4.1.vxa bf=Outz.vx -cp of=compareT4_1.vx ig=InzT4.vx os=compareT4_1.txt
```

```
a=`tail -1 compareT1_1.txt | sed 's/dsc//g'`
b=`tail -1 compareT2_1.txt | sed 's/dsc//g'`
c=`tail -1 compareT3_1.txt | sed 's/dsc//g'`
d=`tail -1 compareT4_1.txt | sed 's/dsc//g'`
new=$a$b$c$d
echo $new >> outputT.txt
```