**Assignment 1:**

**Gesture Recognition**

**Jason Mann**

**2/18/2013**

**COMS 4735: Visual Interfaces for Computers**

**Table of Contents**

**Readme.txt ………………………… 3**

**Application Views…………………4**

# Possible Hand Gestures……...Code and Documentation

Readme.txt

Creator: Jason Carlisle Mann (on2valhalla | jcm2207@columbia.edu)

Updated: 2013.02.18

------------

INTRODUCTION

------------

This project was designed for an assignment in COMS 4735 Visual Interfaces for

Computers under the supervision of Prof. John Kender at Columbia University in

Spring 2013.

The goal of this project was to filter out skin regions in imagery (static or

motion), detect regions belonging to hands, and then interpret specific pre-

defined gestures from the captures, as part of a password grammar.

The project is coded in C++, relying on OpenCV 2.4.3 for computer vision

algorithms, and QT 4.8 (and QT Creator) for the design of the GUI forms. Some

code and inspiration was found in the OpenCV2 Cookbook by Robert Laganiere,

specifically the design of the SkinDetector and Controller classes (though not

necessarily the implementation).

The skin regions are differentiated from the background by manual configuration

of an HSV thresholding mask (setting min and max for each value) through the use

of sliders on an introductory form. While photos can be thresholded, they are

not implemented for use in the password grammar. The masks are applied using

cv::inRange() to provide a binary image of skin regions, and that image is then

blurred and closed (eroded()/dilate()). Once the correct mask is found, you

proceed to the detection form, which begins looking for hand gestures.

Immediately, any faces in the frame are identified and eliminated from

consideration, using OpenCV's built-in Haar cascade classifier

(frontalface\_alt\_tree). Any other skin regions in the frame are considered to be

hands, so a long sleeve shirt is required. These regions are captured as

contours using cv::findContours. The gestures used for the passwords are

captured at regular intervals (hard coded to 3 seconds) to allow the user time to

configure the next gesture.

I chose five gestures (or absence of) for my grammar, a closed fist (FIST),

single finger vertical point (POINT), and closed finger extended palm (PALM), no

detected gestures (NONE), and a skin region that was unidentified (UNK). These

are differentiated based on ratios (width/height) of their respective upright

and minimum fit bounding rectangles, as well as the general mass of the moment

of the contour points (see cv::Moment::m00). Any number of hand gestures can be

captured in the frame, although a maximum of 2 is used for the password grammar.

If there are two gestures in the capture, there is no differentiation between

them, they are viewed as a non-ordered set. The passwords are hard coded into the

PasswordCheck class as a tree of conditional statements, though this could

easily be expanded to a search tree. The password sequences are all 3 sequences

long (although they can have up to two gestures per sequence). There are two

special sequences different from the passwords, two PALMs in one frame, and a

sequence of 3 captures with no gesture detected. Two PALMs signifies that the

password attempt is complete, and to check against the grammar. This check also

occurs automatically if a sequence of 6 uninterrupted captures is encountered. If

the 3 first captures are all NONE or UNK, the grammar restarts with the 4th

capture.

The hard coded accepted password sequences are as follows:

(Capture 1 | Capture 2 | Capture 3)

1. FIST | FIST | FIST

2. POINT | FIST | PALM

3. PALM | (NONE OR UNK) | PALM

4. (PALM & POINT) | (FIST & FIST) | (POINT & FIST)

Two special sequences:

1. (PALM & PALM)

>immediately check the recorded sequence preceding

2. (NONE OR UNK) | (NONE OR UNK) |(NONE OR UNK)

>reset, and check for one of the passwords listed

>above starting with the 4th capture

Upon entering a successful password, the system announces in the text box

PASSWORD ACCEPTED, and on failure: INTRUDER.... INTRUDER....

-----------

ENVIRONMENT

-----------

Due to creating a mask selection form, I tested in multiple locations and was

able to achieve ~70% accuracy in most locations. Those that fared worse were

generally sunlit areas, areas with uni-directional lighting, and areas that

have flesh colored tones, or light birch wood everywhere (like the Science

and Engineering Library). In a controlled environment, I have seen up to ~90%

success, though with a system trained towards my hands. All in all, I attempted

to eliminate the environment concerns from this project.

However, because of the specific geometric data used, a long sleeve shirt is

required for accurate captures. Also, the recommended distance for the built-

in MacBook Air iSight webcam (late 2011) is between 2.5 and 4 feet. This is due

mostly to the mass measurement differing a palm from other gestures.

------

ISSUES

------

1. The masks are configurable with the first form, but if

there are skin-similar (in HSV) regions in the background, it can be hard to

find the right masks. I have added a hardcoded inverse option that allows for a

bit more configuration (specifically if the sought after color bridges the max

and min of the hue spectrum) but there are environments that can still cause

malfunction. This could be avoided by using either background substitution

(Codebook method), or histogram training.

2. The gesture recognition relies on ranges probably specific to my hands,

and will have to be customized to each user in the code. This can be solved by

an introductory training/calibration session for each user, but is beyond the

scope of this project. A more fitting calibration would also be to separately

label the left and right hands.

3. Auto-focus, and auto-exposure on modern web-cameras can cause a problem

with the mask no longer being a valid fit, so if possible disable these

options on your camera. On OSX there is no direct option, though through the

use of iGlasses(Shareware/$19.95) you can disable auto-exposure on the built-

in MacBook web cameras.

4. Currently only supports hardcoded camera number (set at 1 for iGlasses).

to be fixed soon....

------------

ORGANIZATION

------------

Source code specific to this program is contained in src/, extendable classes

are contained in include/ as follows:

GesturePasswords/

GesturePassword.pro -QT Creator project file

README.txt -this file

bin/

|GesturePasswords.tar.gz -binary package compiled 02.18.2013 (OSX)

doc/

|gesture data.xlsx (recorded data for my hand gestures)

include/

|colorhistogram.h -a class for displaying a histogram

| of the image (modified for HSV)

|hand.h

|skindetectcontroller.cpp -a singleton wrapper with

| ease of use interfaces for

|skindetectcontroller.h the SkinDetector class

|skindetector.cpp

|skindetector.h -the main processing class

| for detecting skin regions

src/

|gesturedetector.cpp -detector form functions

|gesturedetector.h -detector form header

|gesturedetector.u -detector form design (QDialog QT 4.8)

|main.cpp -project main method

|maskselection.cpp -mask selection form functions

|maskselection.h -mask selection form header

|maskselection.ui -mask selection form design (QWindow QT 4.8)

|passwordcheck.h -class for storing and processing

| hand gesture password sequences

-----

USAGE

-----

A binary is provided in bin/ that was compiled on 02.18.2013 under

OSX 10.8 with all required dependencies, no knowledge of whether this works

on other systems.

Otherwise, if you have both OpenCV2 and QT Creator, simply import into Creator

and run from there. If you have QT 4.8 you can also use the .pro file to

compile from the command line.

# **Main.cpp**

/\*

Created by: Jason Carlisle Mann (on2valhalla | jcm2207@columbia.edu)

Main method, and start of all the fun

\*/

#include "../src/maskselection.h"

#include <QApplication>

#include <iostream>

int main(int argc, char \*argv[])

{

QApplication a(argc, argv);

MaskSelection w;

w.show();

return a.exec();

}

## MaskSelection.h

/\*------------------------------------------------------------------------------------------\*\

Created by: Jason Carlisle Mann (on2valhalla | jcm2207@columbia.edu)

Some inspiration for design and organization from Robert Laganiere (OpenCV2 Cookbook)

This class is a QT based form, with sliders and a image display area

so that you can actively select a mask to create a binary skin image from

a standard RGB. The primary purpose of the form is to serve video, but

it also has an option to display photos. You select a mask by moving

sliders (in HSV values) for the min and max thresholds sent to the

SkinDetector class

\\*------------------------------------------------------------------------------------------\*/

#ifndef MASKSELECTION\_H

#define MASKSELECTION\_H

#include <QMainWindow>

#include <QFileDialog>

#include <QColorDialog>

#include <QTimer>

//sys

#include <iostream>

#include <string>

//OpenCV

#include <opencv2/core/core.hpp>

#include <opencv2/highgui/highgui.hpp>

//color detector, controller, histogram

#include "../include/skindetectcontroller.h"

#include "../include/colorhistogram.h"

#include "../src/gesturedetector.h"

namespace Ui {

class MaskSelection;

}

class MaskSelection : public QMainWindow{

Q\_OBJECT

public:

MaskSelection(QWidget \*parent = 0);

~MaskSelection();

protected:

void changeEvent(QEvent \*e);

void displayMat(const cv::Mat& img);

// camera vars

cv::VideoCapture cap;

static const int CAMERA = 1;

// timer vars

QTimer\* timer;

bool process;

bool histEnable;

private:

Ui::MaskSelection \*ui;

GestureDetector \*gd;

// the thresholding masks

cv::Scalar min, max;

cv::Mat histogram;

ColorHistogram cHist;

private slots:

void processColorDetection();

void showHistogram();

void setImage();

void toggleCamera();

void updateTimer();

void setThreshold();

void beginDetection();

void setMinHue(int value);

void setMinSat(int value);

void setMinValue(int value);

void setMaxHue(int value);

void setMaxSat(int value);

void setMaxValue(int value);

};

#endif // MASKSELECTION\_H

MaskSelection.cpp

/\*------------------------------------------------------------------------------------------\*\

Created by: Jason Carlisle Mann (on2valhalla | jcm2207@columbia.edu)

Some inspiration for design and organization from Robert Laganiere (OpenCV2 Cookbook)

This class is a QT based form, with sliders and a image display area

so that you can actively select a mask to create a binary skin image from

a standard RGB. The primary purpose of the form is to serve video, but

it also has an option to display photos. You select a mask by moving

sliders (in HSV values) for the min and max thresholds sent to the

SkinDetector class

\\*------------------------------------------------------------------------------------------\*/

#include "../src/maskselection.h"

#include "../src/ui\_maskselection.h"

// Constructor

MaskSelection::MaskSelection(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::MaskSelection)

{

//setup forms

ui->setupUi(this);

//set up video ------------------

//get camera

cap.open(CAMERA);

// check if we succeeded, if not do not enable camera toggle

if(!cap.isOpened())

ui->pushButton\_Camera->setEnabled(false);

//set up timer for camera display

timer = new QTimer(this);

//end setup video ---------------------

// slot action methods -------------

// select color

connect(timer, SIGNAL(timeout()), this, SLOT(updateTimer()));

connect(ui->pushButton\_OpenImage, SIGNAL(clicked()), this, SLOT(setImage()));

connect(ui->pushButton\_Camera, SIGNAL(clicked()), this, SLOT(toggleCamera()));

connect(ui->pushButton\_Process, SIGNAL(clicked()), this, SLOT(processColorDetection()));

connect(ui->pushButton\_Histogram, SIGNAL(clicked()), this, SLOT(showHistogram()));

connect(ui->pushButton\_Detect, SIGNAL(clicked()), this, SLOT(beginDetection()));

connect(ui->verticalSlider\_MinHue, SIGNAL(valueChanged(int)),

this, SLOT(setMinHue(int)));

connect(ui->verticalSlider\_MinSat, SIGNAL(valueChanged(int)),

this, SLOT(setMinSat(int)));

connect(ui->verticalSlider\_MinValue, SIGNAL(valueChanged(int)),

this, SLOT(setMinValue(int)));

connect(ui->verticalSlider\_MaxHue, SIGNAL(valueChanged(int)),

this, SLOT(setMaxHue(int)));

connect(ui->verticalSlider\_MaxSat, SIGNAL(valueChanged(int)),

this, SLOT(setMaxSat(int)));

connect(ui->verticalSlider\_MaxValue, SIGNAL(valueChanged(int)),

this, SLOT(setMaxValue(int)));

//end slots ----------------

//default settings

process = false;

histEnable = false;

cHist = ColorHistogram();

// Hsv default filter (for home environment)

min[0] = 0;

min[1] = 40;

min[2] = 93;

max[0] = 23;

max[1] = 255;

max[2] = 255;

setThreshold();

// set defaults for sliders

ui->verticalSlider\_MinHue->setSliderPosition(min[0]);

ui->verticalSlider\_MinSat->setSliderPosition(min[1]);

ui->verticalSlider\_MinValue->setSliderPosition(min[2]);

ui->verticalSlider\_MaxHue->setSliderPosition(max[0]);

ui->verticalSlider\_MaxSat->setSliderPosition(max[1]);

ui->verticalSlider\_MaxValue->setSliderPosition(max[2]);

}

// Destructor

MaskSelection::~MaskSelection()

{

delete ui;

delete timer;

delete gd;

}

void MaskSelection::changeEvent(QEvent \*e)

{

QMainWindow::changeEvent(e);

switch (e->type()) {

case QEvent::LanguageChange:

ui->retranslateUi(this);

break;

default:

break;

}

}

/\*

This method is called by a button on the form

and in turn pulls up an open file dialog, where

the user can select a photo for thresholding

\*/

void MaskSelection::setImage()

{

timer->stop();

QFileDialog::Options options;

QString selectedFilter;

QString fileName = QFileDialog::getOpenFileName(this,

tr("Open Image Files"),

"",

tr("Image files (\*.jpg \*.jpeg \*.png \*.gif \*.bmp)"),

&selectedFilter,

options);

if (!fileName.isEmpty()){

cv::Mat image = cv::imread(fileName.toStdString(),1); //0 for grayscale

displayMat(image);

//Send Filename to the skin detector

SkinDetectController::getInstance()->setInputImage(fileName.toStdString());

}

}

/\*

Called by the Camera button, toggles the video feed

on and off by toggling the timer. It will only attempt to

turn on the timer if the cv::VideoCapture is opened

\*/

void MaskSelection::toggleCamera()

{

if(!cap.isOpened())

return;

if(timer->isActive())

timer->stop();

else

{

timer->start(25);

}

}

/\*

A simple conversion method to display a cv::Mat image (BGR or binary)

in a QLabel, that only takes a QImage as a pixelmap.

\*/

void MaskSelection::displayMat(const cv::Mat& image)

{

//BGR openCV Mat to QImage

QImage img\_qt = QImage((const unsigned char\*)image.data,image.cols, image.rows, image.step, QImage::Format\_RGB888);

//For Binary Images

if (img\_qt.isNull()){

//ColorTable for Binary Images

QVector<QRgb> colorTable;

for (int i = 0; i < 256; i++)

colorTable.push\_back(qRgb(i, i, i));

img\_qt = QImage((const unsigned char\*)image.data,image.cols, image.rows, image.step, QImage::Format\_Indexed8);

img\_qt.setColorTable(colorTable);

}

//Display the QImage in the Label

QPixmap img\_pix = QPixmap::fromImage(img\_qt.rgbSwapped()); //BGR to RGB

this->ui->label->setPixmap(img\_pix.scaled(ui->label->size(), Qt::KeepAspectRatio));

}

/\*

Called by the process button, this method toggles the processing

and the display of the processed images/video. This method calls

the process function from the SkinDetector class

\*/

void MaskSelection::processColorDetection()

{

if(timer->isActive())

process = !process;

else

{

SkinDetectController::getInstance()->process();

cv::Mat resulting = SkinDetectController::getInstance()->getLastResult();

if (!resulting.empty())

displayMat(resulting);

}

}

/\*

Controls the display of the video, on the interval set in

toggleCamera above. It also regulates the display of the

histogram, and processing of the video.

\*/

void MaskSelection::updateTimer()

{

cv::Mat img;

cap >> img; //capture a frame

//send SkinDetector the frame

SkinDetectController::getInstance()->setInputImage(img);

if(histEnable)

{ // update the histogram

histogram = cHist.getHistogramImage(img);

cv::imshow("Histogram", histogram);

}

if(process)

{ //process the frame

SkinDetectController::getInstance()->process();

//retrieve the processed frame

cv::Mat result = SkinDetectController::getInstance()->getLastResult();

if (!result.empty())

img = result;

}

displayMat(img);

}

/\*

Toggles the histogram window on and off

\*/

void MaskSelection::showHistogram()

{

if(timer->isActive() && !histEnable)

{ //create histogram window for video display

cv::namedWindow("Histogram", cv::WINDOW\_AUTOSIZE);

histEnable = true;

}

else if (!timer->isActive())

{ //create histogram for image display

histogram = cHist.getHistogramImage(SkinDetectController::getInstance()->getHSVImage());

cv::imshow("Histogram", histogram);

histEnable = false;

}

else

{ //destroy histogram window

cv::destroyWindow("Histogram");

histEnable = false;

}

}

/\*

Sets the skin detector's masks min and max from

their cached values

\*/

void MaskSelection::setThreshold()

{

SkinDetectController::getInstance()->setThreshold(min, max);

if(!timer->isActive() && process)

processColorDetection();

}

/\*

Changes minHue from slider

\*/

void MaskSelection::setMinHue(int value)

{

min[0] = value;

this->ui->label\_MinHue->setText(QString::number(value));

setThreshold();

}

/\*

Changes minimum saturation from slider

\*/

void MaskSelection::setMinSat(int value)

{

min[1] = value;

this->ui->label\_MinSat->setText(QString::number(value));

setThreshold();

}

/\*

Changes minimum value from slider

\*/

void MaskSelection::setMinValue(int value)

{

min[2] = value;

this->ui->label\_MinValue->setText(QString::number(value));

setThreshold();

}

/\*

Changes maximum hue from slider

\*/

void MaskSelection::setMaxHue(int value)

{

max[0] = value;

this->ui->label\_MaxHue->setText(QString::number(value));

setThreshold();

}

/\*

Changes maximum saturation from slider

\*/

void MaskSelection::setMaxSat(int value)

{

max[1] = value;

this->ui->label\_MaxSat->setText(QString::number(value));

setThreshold();

}

/\*

Changes maximum value from slider

\*/

void MaskSelection::setMaxValue(int value)

{

max[2] = value;

this->ui->label\_MaxValue->setText(QString::number(value));

setThreshold();

}

/\*

Creates a new detection form if necessary,

and sets up the form to begin recording hand

gestures

\*/

void MaskSelection::beginDetection()

{

if(!gd)

gd = new GestureDetector(this);

gd->show();

if(cap.isOpened())

{

toggleCamera();

gd->setCap(cap);

gd->start();

}

}

SkinDetectController.h

/\*

Modified from ColorDetector from chapter 3 of the cookbook:

Computer Vision Programming using the OpenCV Library.

by Robert Laganiere, Packt Publishing, 2011.

Created by: Jason Carlisle Mann (on2valhalla | jcm2207@columbia.edu)

A singleton wrapper class for SkinDetector, to gain continuity

throughout the program, and also to provide more simple access

to its methods, and a local cache of the most recent images.

\*/

#if !defined SKN\_CNTRLLR

#define SKN\_CNTRLLR

#include <opencv2/core/core.hpp>

#include <opencv2/highgui/highgui.hpp>

#include <iostream>

#include "../include/skindetector.h"

class SkinDetectController

{

private:

static SkinDetectController \*singleton;

SkinDetector \*sknDetect;

// hsvImage storage

cv::Mat hsvImage;

cv::Mat resultImg;

public:

SkinDetectController()

{

sknDetect = new SkinDetector();

}

// Deletes all processor objects created by the controller.

~SkinDetectController()

{

delete sknDetect;

}

// Singleton static members

static SkinDetectController \*getInstance()

{

if (singleton == 0)

singleton= new SkinDetectController;

return singleton;

}

// Releases the singleton instance of this controller.

static void destroy() {

if (singleton != 0)

{

delete singleton;

singleton= 0;

}

}

// Sets the input hsvImage. Reads it from file.

bool setInputImage(std::string filename)

{

cv::Mat imgIn = cv::imread(filename);

// convert color space

cv::cvtColor(imgIn, hsvImage, CV\_BGR2HSV);

if (!hsvImage.data)

return false;

else

return true;

}

// Sets the input hsvImage.

bool setInputImage(cv::Mat imgIn) {

// convert color space

cv::cvtColor(imgIn, hsvImage, CV\_BGR2HSV);

if (!hsvImage.data)

return false;

else

return true;

}

cv::Mat getInputImage(){

cv::Mat imgOut;

cv::cvtColor(hsvImage, imgOut, CV\_HSV2BGR);

return imgOut;

}

// Returns the current input hsvImage.

// NOTE: this returns HSV!!!!!

const cv::Mat getHSVImage() const {

return hsvImage;

}

const cv::Mat getLastResult() const{

return resultImg;

}

void setThreshold(cv::Scalar min, cv::Scalar max){

sknDetect->setThreshold(min, max);

}

void process(){

resultImg = sknDetect->processHSV(hsvImage);

}

};

#endif

SkinDetectController.cpp

/\*

Modified from ColorDetector from chapter 3 of the cookbook:

Computer Vision Programming using the OpenCV Library.

by Robert Laganiere, Packt Publishing, 2011.

Created by: Jason Carlisle Mann (on2valhalla | jcm2207@columbia.edu)

A singleton wrapper class for SkinDetector, to gain continuity

throughout the program, and also to provide more simple access

to its methods, and a local cache of the most recent images.

\*/

#include "../include/skindetectcontroller.h"

SkinDetectController \*SkinDetectController::singleton=0;

SkinDetector.h

/\*

Modified from ColorDetector from chapter 3 of the cookbook:

Computer Vision Programming using the OpenCV Library.

by Robert Laganiere, Packt Publishing, 2011.

Created by: Jason Carlisle Mann (on2valhalla | jcm2207@columbia.edu)

This class uses holds input threshold min and max

masks to process an image for skin blobs in HSV colorspace

\*/

#if !defined SKINDETECT

#define SKINDETECT

#include <opencv2/core/core.hpp>

#include <opencv2/imgproc/imgproc.hpp>

#include <iostream>

class SkinDetector

{

private:

// HSV min and max limits as array of Scalars

cv::Scalar hsvThreshold[2];

// image containing result of processing

cv::Mat resultImg;

// image containing converted color space

cv::Mat converted;

public:

//empty Constructor

SkinDetector()

{

// initialize default parameters

hsvThreshold[0][0] = 0;

hsvThreshold[0][1] = 0;

hsvThreshold[0][2] = 0;

hsvThreshold[1][0] = 180;

hsvThreshold[1][1] = 255;

hsvThreshold[1][2] = 255;

}

void setThreshold(cv::Scalar min, cv::Scalar max)

{

hsvThreshold[0] = min;

hsvThreshold[1] = max;

//std::cout << "min: " << min << "\t" << "max" << max << "\n";

}

void getThreshold(cv::Scalar &min, cv::Scalar &max)

{

min = hsvThreshold[0];

max = hsvThreshold[1];

}

// Processes an already HSV image. Returns a 1-channel binary image.

cv::Mat processHSV(const cv::Mat &image);

};

#endif

SkinDetector.cpp

/\*

Modified from ColorDetector from chapter 3 of the cookbook:

Computer Vision Programming using the OpenCV Library.

by Robert Laganiere, Packt Publishing, 2011.

Created by: Jason Carlisle Mann (on2valhalla | jcm2207@columbia.edu)

This class uses holds input threshold min and max

masks to process an image for skin blobs in HSV colorspace

\*/

#include "../include/skindetector.h"

#include "../include/colorhistogram.h"

/\*

Processes an HSV image and returns a binary image

containing blobs of skin regions.

@hsvImg input HSV colorspace OpenCV2 image

@return processed binary blob image

\*/

cv::Mat SkinDetector::processHSV(const cv::Mat &hsvImg)

{

//re-allocate binary map if necessary

//if so create one channel image with

//same cols and rows as original

resultImg.create(hsvImg.rows, hsvImg.cols, CV\_8U);

//reduce the colors for faster processing

ColorHistogram h;

h.colorReduce(hsvImg, 12);

//threshold the image with the stored masks

cv::inRange(hsvImg, hsvThreshold[0], hsvThreshold[1], resultImg);

//filtering parameter, increase size for greater effect

cv::Mat morpElement(5,5,CV\_8U,cv::Scalar(1));

//current morphological processing functions to

//close the skin blobs

cv::erode(resultImg, resultImg, morpElement);

cv::blur(resultImg, resultImg, morpElement.size());

cv::dilate(resultImg, resultImg, morpElement);

//optional morphological processing is helpful

//depending on the background environment

// cv::bitwise\_not(resultImg, resultImg);

// cv::erode(resultImg, resultImg, morpElement);

// cv::dilate(resultImg, resultImg, morpElement);

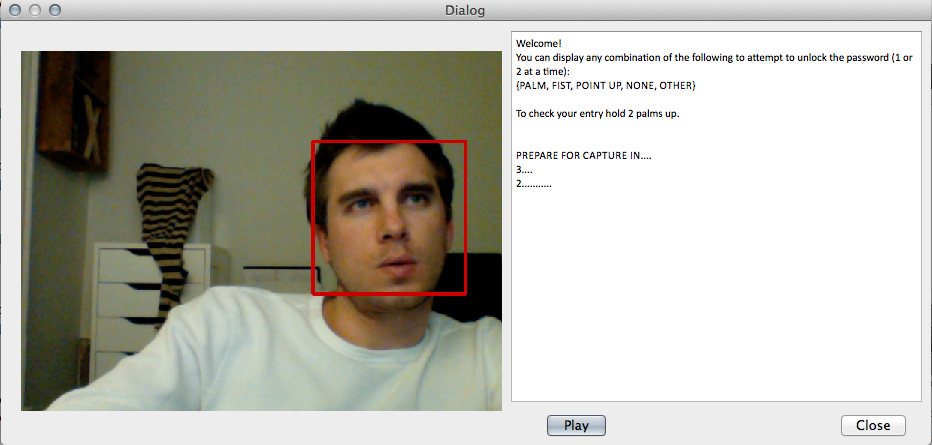
// cv::morphologyEx(resultImg,resultImg,cv::MORPH\_CLOSE, morpElement);

return resultImg;

}

Application Views (3 forms: Mask Selection, Histogram, Gesture Detection)

|  |
| --- |
| Enki_Files:Documents:school:Visual Interfaces:projects:GesturePasswords:doc:img:Screen Shot 2013-02-19 at 2.25.47 AM.png |
| The mask selection form allowsEnki_Files:Documents:school:Visual Interfaces:projects:GesturePasswords:doc:img:Screen Shot 2013-02-19 at 2.22.17 AM.png you to see your mask settings immediately take effect |
|  |



|  |  |
| --- | --- |
| Enki_Files:Documents:school:Visual Interfaces:projects:GesturePasswords:doc:img:2:pass_2_2.jpg | Enki_Files:Documents:school:Visual Interfaces:projects:GesturePasswords:doc:img:2:pass_2_1.jpg |
| Enki_Files:Documents:school:Visual Interfaces:projects:GesturePasswords:doc:img:2:pass_2_0.jpg | Enki_Files:Documents:school:Visual Interfaces:projects:GesturePasswords:doc:img:4:pass_4_1.jpg |
| Enki_Files:Documents:school:Visual Interfaces:projects:GesturePasswords:doc:img:2:pass_2_3.jpg | Enki_Files:Documents:school:Visual Interfaces:projects:GesturePasswords:doc:img:5:pass_0_0.jpg |

# Possible Hand Gestures

**-Closed fist, closed palm, vertical single finger point, or none**

**-One or Two gestures at a time**

# 10 Example Sequences:

**-5 correctly read sequences with the right password, accepted: PASS**

**-2 correctly read, wrong password, denied: PASS**

**-2 incorrectly read, right password, denied: FAIL**

**-1 incorrectly read, wrong password, accepted: FAIL**

|  |  |
| --- | --- |
| Enki_Files:Documents:school:Visual Interfaces:projects:GesturePasswords:doc:img:1:pass_0_0.jpg | Sequence 1**-Correct read** **-Correct password** PREPARE FOR CAPTURE IN....  3...........  2...........  1...........  FIST area: 5727 bratio: 1.361 rratio: 0.707 mratio: 1.638  PREPARE FOR CAPTURE IN....  3...........  2...........  1...........  FIST area: 5823.5 bratio: 1.354 rratio: 1.359 mratio: 1.452  PREPARE FOR CAPTURE IN....  3...........  2...........  1...........  FIST area: 5861 bratio: 1.333 rratio: 1.337 mratio: 1.493  PREPARE FOR CAPTURE IN....  3...........  2...........  1...........  PALM area: 11449 bratio: 0.656 rratio: 0.567 mratio: 1.918  and: PALM area: 13087.5 bratio: 0.791 rratio: 1.587 mratio: 2.442  ----------------------  PASSWORD ACCEPTED!!!!!!  ---------------------- |
| Enki_Files:Documents:school:Visual Interfaces:projects:GesturePasswords:doc:img:1:pass_0_1.jpg |
| Enki_Files:Documents:school:Visual Interfaces:projects:GesturePasswords:doc:img:1:pass_0_2.jpg |
| Enki_Files:Documents:school:Visual Interfaces:projects:GesturePasswords:doc:img:1:pass_0_3.jpg |

|  |  |
| --- | --- |
| Enki_Files:Documents:school:Visual Interfaces:projects:GesturePasswords:doc:img:2:pass_2_0.jpg | Sequence 2 **-Correct read** **-Correct password** PREPARE FOR CAPTURE IN....  3...........  2...........  1...........  POINT area: 6920 bratio: 0.638 rratio: 0.636 mratio: 2.131  PREPARE FOR CAPTURE IN....  3...........  2...........  1...........  FIST area: 7303.5 bratio: 1.089 rratio: 1.109 mratio: 1.871  PREPARE FOR CAPTURE IN....  3...........  2...........  1...........  PALM area: 14128 bratio: 0.760 rratio: 1.752 mratio: 2.628  PREPARE FOR CAPTURE IN....  3...........  2...........  1...........  PALM area: 11085.5 bratio: 0.638 rratio: 0.562 mratio: 1.804  and: PALM area: 14062 bratio: 0.750 rratio: 1.767 mratio: 2.581  ----------------------  PASSWORD ACCEPTED!!!!!!  ---------------------- |
| Enki_Files:Documents:school:Visual Interfaces:projects:GesturePasswords:doc:img:2:pass_2_1.jpg |
| Enki_Files:Documents:school:Visual Interfaces:projects:GesturePasswords:doc:img:2:pass_2_2.jpg |
| Enki_Files:Documents:school:Visual Interfaces:projects:GesturePasswords:doc:img:2:pass_2_3.jpg |

|  |  |
| --- | --- |
| Enki_Files:Documents:school:Visual Interfaces:projects:GesturePasswords:doc:img:3:pass_3_0.jpg | Sequence 3 **-Correct read** **-Correct password** PREPARE FOR CAPTURE IN....  3...........  2...........  1...........  PALM area: 12934.5 bratio: 0.727 rratio: 1.843 mratio: 2.622  PREPARE FOR CAPTURE IN....  3...........  2...........  1...........  NONE area: 0 bratio: 0.000 rratio: 0.000 mratio: 0.000  PREPARE FOR CAPTURE IN....  3...........  2...........  1...........  PALM area: 12499 bratio: 0.703 rratio: 1.696 mratio: 2.137  PREPARE FOR CAPTURE IN....  3...........  2...........  1...........  PALM area: 11306 bratio: 0.585 rratio: 0.583 mratio: 1.393  and: PALM area: 12570 bratio: 0.722 rratio: 1.688 mratio: 2.253  ----------------------  PASSWORD ACCEPTED!!!!!!  ---------------------- |
| Enki_Files:Documents:school:Visual Interfaces:projects:GesturePasswords:doc:img:3:pass_3_1.jpg |
| Enki_Files:Documents:school:Visual Interfaces:projects:GesturePasswords:doc:img:3:pass_3_2.jpg |
| Enki_Files:Documents:school:Visual Interfaces:projects:GesturePasswords:doc:img:3:pass_3_3.jpg |

|  |  |
| --- | --- |
| Enki_Files:Documents:school:Visual Interfaces:projects:GesturePasswords:doc:img:4:pass_4_0.jpg | Sequence 4 **-Correct read** **-Correct password** PREPARE FOR CAPTURE IN....  3...........  2...........  1...........  NONE area: 0 bratio: 0.000 rratio: 0.000 mratio: 0.000  PREPARE FOR CAPTURE IN....  3...........  2...........  1...........  NONE area: 0 bratio: 0.000 rratio: 0.000 mratio: 0.000  PREPARE FOR CAPTURE IN....  3...........  2...........  1...........  NONE area: 0 bratio: 0.000 rratio: 0.000 mratio: 0.000 |
| Enki_Files:Documents:school:Visual Interfaces:projects:GesturePasswords:doc:img:4:pass_4_1.jpg |
| Enki_Files:Documents:school:Visual Interfaces:projects:GesturePasswords:doc:img:4:pass_4_2.jpg |
| Enki_Files:Documents:school:Visual Interfaces:projects:GesturePasswords:doc:img:4:pass_4_3.jpg | Sequence 4 (cont.) PREPARE FOR CAPTURE IN....  3...........  2...........  1...........  FIST area: 6602.5 bratio: 1.261 rratio: 1.264 mratio: 1.479  PREPARE FOR CAPTURE IN....  3...........  2...........  1...........  FIST area: 6659.5 bratio: 1.264 rratio: 1.267 mratio: 1.437  PREPARE FOR CAPTURE IN....  3...........  2...........  1...........  FIST area: 6650.5 bratio: 1.208 rratio: 0.803 mratio: 1.674  PREPARE FOR CAPTURE IN....  3...........  2...........  1...........  PALM area: 11708.5 bratio: 0.657 rratio: 0.577 mratio: 1.779  and: PALM area: 13538.5 bratio: 0.777 rratio: 1.676 mratio: 2.556  ----------------------  PASSWORD ACCEPTED!!!!!!  ---------------------- |
| Enki_Files:Documents:school:Visual Interfaces:projects:GesturePasswords:doc:img:4:pass_4_4.jpg |
| Enki_Files:Documents:school:Visual Interfaces:projects:GesturePasswords:doc:img:4:pass_4_5.jpg |
| Enki_Files:Documents:school:Visual Interfaces:projects:GesturePasswords:doc:img:4:pass_4_6.jpg |

|  |  |
| --- | --- |
| Enki_Files:Documents:school:Visual Interfaces:projects:GesturePasswords:doc:img:5:pass_0_0.jpg | Sequence 5 ****-Correct read**** ****-Correct password**** PREPARE FOR CAPTURE IN....  3....  2...........  1...........  POINT area: 6152.5 bratio: 0.660 rratio: 0.657 mratio: 2.131  and: PALM area: 11655 bratio: 0.695 rratio: 0.563 mratio: 2.086  PREPARE FOR CAPTURE IN....  3...........  2...........  1...........  FIST area: 6540.5 bratio: 1.077 rratio: 1.078 mratio: 1.364  and: FIST area: 5740 bratio: 1.310 rratio: 1.347 mratio: 1.610  PREPARE FOR CAPTURE IN....  3...........  2...........  1...........  FIST area: 6479.5 bratio: 1.100 rratio: 1.101 mratio: 1.375  and: POINT area: 5740 bratio: 0.651 rratio: 0.615 mratio: 2.416  PREPARE FOR CAPTURE IN....  3...........  2...........  1...........  PALM area: 11484.5 bratio: 0.672 rratio: 0.571 mratio: 1.896  and: PALM area: 11762.5 bratio: 0.785 rratio: 1.566 mratio: 2.309  ----------------------  PASSWORD ACCEPTED!!!!!!  ---------------------- |
| Enki_Files:Documents:school:Visual Interfaces:projects:GesturePasswords:doc:img:5:pass_0_1.jpg |
| Enki_Files:Documents:school:Visual Interfaces:projects:GesturePasswords:doc:img:5:pass_0_2.jpg |
| Enki_Files:Documents:school:Visual Interfaces:projects:GesturePasswords:doc:img:5:pass_0_3.jpg |

|  |  |
| --- | --- |
| Enki_Files:Documents:school:Visual Interfaces:projects:GesturePasswords:doc:img:6:pass_1_0.jpg | Sequence 6 **-Correct read**  **-Incorrect password**  PREPARE FOR CAPTURE IN....  3...........  2...........  1...........  POINT area: 6333 bratio: 0.711 rratio: 0.672 mratio: 2.592  PREPARE FOR CAPTURE IN....  3...........  2...........  1...........  POINT area: 6303 bratio: 0.631 rratio: 0.628 mratio: 2.222  PREPARE FOR CAPTURE IN....  3...........  2...........  1...........  POINT area: 6443 bratio: 0.707 rratio: 0.665 mratio: 2.705  PREPARE FOR CAPTURE IN....  3...........  2...........  1...........  PALM area: 11158.5 bratio: 0.596 rratio: 0.594 mratio: 1.385  and: PALM area: 12773.5 bratio: 0.780 rratio: 1.652 mratio: 2.443  ----------------------  INTRUDER... INTRUDER....  ---------------------- |
| Enki_Files:Documents:school:Visual Interfaces:projects:GesturePasswords:doc:img:6:pass_1_1.jpg |
| Enki_Files:Documents:school:Visual Interfaces:projects:GesturePasswords:doc:img:6:pass_1_2.jpg |
| Enki_Files:Documents:school:Visual Interfaces:projects:GesturePasswords:doc:img:6:pass_1_3.jpg |

|  |  |
| --- | --- |
| Enki_Files:Documents:school:Visual Interfaces:projects:GesturePasswords:doc:img:7:pass_5_0.jpg | Sequence 7 **-Correct read**  **-Incorrect password**  PREPARE FOR CAPTURE IN....  3...........  2...........  1...........  POINT area: 7796 bratio: 0.651 rratio: 0.620 mratio: 2.300  and: PALM area: 14043 bratio: 0.542 rratio: 0.539 mratio: 1.422  PREPARE FOR CAPTURE IN....  3...........  2...........  1...........  FIST area: 8491.5 bratio: 0.921 rratio: 1.089 mratio: 1.721  and: FIST area: 8296.5 bratio: 0.969 rratio: 0.963 mratio: 1.913  PREPARE FOR CAPTURE IN....  3...........  2...........  1...........  PALM area: 13342.5 bratio: 0.716 rratio: 0.588 mratio: 2.082  and: POINT area: 7518.5 bratio: 0.654 rratio: 0.652 mratio: 2.116  PREPARE FOR CAPTURE IN....  3...........  2...........  1...........  PALM area: 12941 bratio: 0.596 rratio: 0.594 mratio: 1.348  and: PALM area: 15713 bratio: 0.734 rratio: 1.934 mratio: 2.780  ----------------------  INTRUDER... INTRUDER....  ---------------------- |
| Enki_Files:Documents:school:Visual Interfaces:projects:GesturePasswords:doc:img:7:pass_5_1.jpg |
| Enki_Files:Documents:school:Visual Interfaces:projects:GesturePasswords:doc:img:7:pass_5_2.jpg |
| Enki_Files:Documents:school:Visual Interfaces:projects:GesturePasswords:doc:img:7:pass_5_3.jpg |

|  |  |
| --- | --- |
| Enki_Files:Documents:school:Visual Interfaces:projects:GesturePasswords:doc:img:false positive:pass_1_0.jpg | Sequence 8\*\*\* **-Incorrect read**  **-False Positive Password**  **-Because of the metrics used, similar proportioned gestures will register a false positive.**  PREPARE FOR CAPTURE IN....  3...........  2...........  1...........  FIST area: 5569 bratio: 1.067 rratio: 1.067 mratio: 1.551  PREPARE FOR CAPTURE IN....  3...........  2...........  1...........  UNK area: 5392.5 bratio: 1.050 rratio: 1.065 mratio: 1.985  PREPARE FOR CAPTURE IN....  3...........  2...........  1...........  POINT area: 9024.5 bratio: 0.627 rratio: 0.624 mratio: 1.400  and: PALM area: 10638 bratio: 0.766 rratio: 1.843 mratio: 2.910  PREPARE FOR CAPTURE IN....  3...........  2...........  1...........  POINT area: 9174.5 bratio: 0.634 rratio: 0.631 mratio: 1.393  and: PALM area: 10668 bratio: 0.783 rratio: 1.863 mratio: 3.026  PREPARE FOR CAPTURE IN....  3...........  2...........  1...........  PALM area: 9004 bratio: 0.660 rratio: 0.574 mratio: 1.854  and: PALM area: 10273.5 bratio: 0.757 rratio: 1.676 mratio: 2.414  ----------------------  PASSWORD ACCEPTED!!!!!!  ---------------------- |
| Enki_Files:Documents:school:Visual Interfaces:projects:GesturePasswords:doc:img:false positive:pass_1_1.jpg |
| Enki_Files:Documents:school:Visual Interfaces:projects:GesturePasswords:doc:img:false positive:pass_1_2.jpg |
| Enki_Files:Documents:school:Visual Interfaces:projects:GesturePasswords:doc:img:false positive:pass_1_4.jpg |
|  |

|  |  |
| --- | --- |
| Enki_Files:Documents:school:Visual Interfaces:projects:GesturePasswords:doc:img:false negative 1:pass_1_1.jpg | Sequence 9\*\*\* **-Incorrect read**  **-False Negative Password**  **-Too much skin from the forearm corrupted the read of the fist, causing it to not accept the correct password.**  PREPARE FOR CAPTURE IN....  3...........  2...........  1...........  UNK area: 11727.5 bratio: 0.846 rratio: 0.845 mratio: 1.601  PREPARE FOR CAPTURE IN....  3...........  2...........  1...........  UNK area: 10081 bratio: 0.863 rratio: 0.862 mratio: 1.825  PREPARE FOR CAPTURE IN....  3...........  2...........  1...........  UNK area: 10019 bratio: 0.879 rratio: 0.869 mratio: 1.948  PREPARE FOR CAPTURE IN....  3...........  2...........  1...........  PALM area: 13215 bratio: 0.686 rratio: 0.597 mratio: 1.893  and: PALM area: 15177.5 bratio: 0.774 rratio: 1.697 mratio: 2.490  ----------------------  INTRUDER... INTRUDER....  ---------------------- |
| Enki_Files:Documents:school:Visual Interfaces:projects:GesturePasswords:doc:img:false negative 1:pass_1_2.jpg |
| Enki_Files:Documents:school:Visual Interfaces:projects:GesturePasswords:doc:img:false negative 1:pass_1_3.jpg |
| Enki_Files:Documents:school:Visual Interfaces:projects:GesturePasswords:doc:img:false negative 1:pass_1_4.jpg |

|  |  |
| --- | --- |
| Enki_Files:Documents:school:Visual Interfaces:projects:GesturePasswords:doc:img:false negative 2:pass_2_0.jpg | Sequence 10\*\*\* **-Incorrect read**  **-False Negative Password**  **-The program incorrectly judged a fist, palm, and then a set of two palms, until it ran up to the 6 sequence limit.**  PREPARE FOR CAPTURE IN....  3...........  2...........  1...........  POINT area: 5140.5 bratio: 0.836 rratio: 1.969 mratio: 5.809  and: UNK area: 13140 bratio: 0.621 rratio: 0.618 mratio: 1.430  PREPARE FOR CAPTURE IN....  3...........  2...........  1...........  FIST area: 6459.5 bratio: 1.208 rratio: 1.271 mratio: 1.908  and: FIST area: 6783 bratio: 1.151 rratio: 0.806 mratio: 1.907  PREPARE FOR CAPTURE IN....  3...........  2...........  1...........  UNK area: 6809 bratio: 1.173 rratio: 0.775 mratio: 2.084  and: POINT area: 6623 bratio: 0.645 rratio: 0.643 mratio: 2.340  PREPARE FOR CAPTURE IN....  3...........  2...........  1...........  UNK area: 12254 bratio: 0.655 rratio: 0.614 mratio: 1.619  and: PALM area: 13339.5 bratio: 0.768 rratio: 1.644 mratio: 2.374  PREPARE FOR CAPTURE IN....  3...........  2...........  1...........  UNK area: 12214 bratio: 0.642 rratio: 0.640 mratio: 1.379  and: PALM area: 13356.5 bratio: 0.771 rratio: 1.609 mratio: 2.333  PREPARE FOR CAPTURE IN....  3...........  2...........  1...........  PALM area: 12345 bratio: 0.667 rratio: 1.673 mratio: 1.750  and: UNK area: 13357.5 bratio: 0.644 rratio: 0.642 mratio: 1.459  ----------------------  INTRUDER... INTRUDER....  ---------------------- |
| Enki_Files:Documents:school:Visual Interfaces:projects:GesturePasswords:doc:img:false negative 2:pass_2_1.jpg |
| Enki_Files:Documents:school:Visual Interfaces:projects:GesturePasswords:doc:img:false negative 2:pass_2_2.jpg |
| Enki_Files:Documents:school:Visual Interfaces:projects:GesturePasswords:doc:img:false negative 2:pass_2_3.jpg |
| Enki_Files:Documents:school:Visual Interfaces:projects:GesturePasswords:doc:img:false negative 2:pass_2_4.jpg |
| Enki_Files:Documents:school:Visual Interfaces:projects:GesturePasswords:doc:img:false negative 2:pass_2_5.jpg |