using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using Emgu.CV.UI;

using Emgu.CV;

using Emgu.CV.Structure;

using Emgu.CV.CvEnum;

using Emgu.CV.VideoSurveillance;

using System.IO;

using System.Xml;

using System.Runtime.InteropServices;

using System.Security.Principal;

using Microsoft.Win32.SafeHandles;

namespace OpenCV

{

public partial class frmFinger : Form

{

#region Global Variables

//OpenCV Variables-------------------------------------------------------------------------

MCvFont font = new MCvFont(FONT.CV\_FONT\_HERSHEY\_COMPLEX, 2, 2);

Capture vidCapture;

Image<Bgr, Byte> imgLive;

Image<Bgr, Byte> imgMain;

double MinimumDistanceDepthPointToEndPointRatio = 0.15; // adjust this to find best distance to detect finger

Rectangle FingerBoundingBox;

CircleF PalmBoundingCircle;

PointF[] PalmPointsCollection;

PointF[] FingerPointsCollection;

//Mouse CTRL Variables---------------------------------------------------------------------

[DllImport("user32.dll")]

private static extern void mouse\_event(UInt32 dwFlags, UInt32 dx, UInt32 dy, UInt32 dwData, IntPtr dwExtraInfo);

//Application Variables--------------------------------------------------------------------

int[] X = new int[2];

int[] Y = new int[2];

int MX, MY;

int LX, LY;

int RX, RY;

int BTN = -1;

Boolean fClick = false;

Boolean fDrag = false;

int FrameCount = 0;

#endregion

public frmFinger()

{

InitializeComponent();

}

//Cam//////////////////////////////////////////////////////////////////////////////////////

private void btnCamStart\_Click(object sender, EventArgs e)

{

vidCapture = new Capture();

Application.Idle += new EventHandler(LiveCam);

}

//-----------------------------------------------------------------------------------------

private void btnCamStop\_Click(object sender, EventArgs e)

{

Application.Idle -= new EventHandler(LiveCam);

if (vidCapture != null)

{

vidCapture.Dispose();

}

}

//-----------------------------------------------------------------------------------------

void LiveCam(object sender, EventArgs e)

{

//Get the current frame form capture device

imgLive = vidCapture.QuerySmallFrame().Resize(320, 240, Emgu.CV.CvEnum.INTER.CV\_INTER\_CUBIC).Flip(FLIP.HORIZONTAL);

if(imgLive != null)

{

picCam.Image = imgLive.ToBitmap();

}

}

//\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\

//Processing///////////////////////////////////////////////////////////////////////////////

private void btnProcessStart\_Click(object sender, EventArgs e)

{

txtSysMsg.Text = "Processing Started." + Environment.NewLine + txtSysMsg.Text;

Application.Idle += new EventHandler(HandGestureProcessing);

}

//-----------------------------------------------------------------------------------------

private void btnProcessStop\_Click(object sender, EventArgs e)

{

Application.Idle -= new EventHandler(HandGestureProcessing);

txtSysMsg.Text = "Processing Stopped." + Environment.NewLine + txtSysMsg.Text;

}

//\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\

//Image Processing/////////////////////////////////////////////////////////////////////////

void HandGestureProcessing(object sender, EventArgs e)

{

imgMain = imgLive.Copy();

Image<Gray, Byte> imgSkin = DetectSkinYCC(imgMain, new Ycc((int)HMin.Value, (int)SMin.Value, (int)VMin.Value), new Ycc((int)HMax.Value, (int)SMax.Value, (int)VMax.Value)); //best method

//Image<Gray, Byte> imgSkin = DetectSkinHSV(imgMain, new Hsv((int)HMin.Value, (int)SMin.Value, (int)VMin.Value), new Hsv((int)HMax.Value, (int)SMax.Value, (int)VMax.Value)); //best method

picIP1.Image = imgSkin.ToBitmap();

//use this to eliminate noise-------------

imgSkin = imgSkin.Erode((int)Erode.Value);

picIP2.Image = imgSkin.ToBitmap();

imgSkin = imgSkin.Dilate((int)Dilate.Value);

picIP3.Image = imgSkin.ToBitmap();

//----------------------------------------

//blur the image & threshold---------------

imgSkin = imgSkin.SmoothGaussian((int)Smooth.Value);

picIP4.Image = imgSkin.ToBitmap();

imgSkin = imgSkin.ThresholdBinary(new Gray((int)Threshold.Value), new Gray(255));

picIP5.Image = imgSkin.ToBitmap();

//-----------------------------------------

Extract\_Contour\_Hull\_Defects(imgSkin);

Draw\_Gesture\_Features();

//analysis------------------------------------------------------------------------------------

#region drawing lines from PalmBoundingCircle center to fingertip

int FingerCount = 0;

for (int i = 0; i < FingerPointsCollection.Count(); i++)

{

if (FingerPointsCollection[i].Y < PalmBoundingCircle.Center.Y + PalmBoundingCircle.Radius / 2)

{

int Distance = (int)Math.Sqrt(Math.Pow(FingerPointsCollection[i].X - PalmBoundingCircle.Center.X, 2) + Math.Pow(FingerPointsCollection[i].Y - PalmBoundingCircle.Center.Y, 2));

if (Distance > PalmBoundingCircle.Radius \* 1.25)

{

imgMain.Draw(new LineSegment2DF(FingerPointsCollection[i], PalmBoundingCircle.Center), new Bgr(Color.Yellow), 1);

X[FingerCount] = (int)FingerPointsCollection[i].X;

Y[FingerCount] = (int)FingerPointsCollection[i].Y;

FingerCount++;

if (FingerCount == 2) break;

}

}

}

imgMain.Draw(FingerCount.ToString(), ref font, new Point(30, 50), new Bgr(Color.Red));

#endregion

DevelopGride();

picIP6.Image = imgMain.ToBitmap();

if (FingerCount == 0)

{

lblCMD.Text = "NO";

return;

}

//logical programming--------------------

if (FingerCount == 1)

{

MX = X[0]; MY = Y[0];//geting the Middle btn

MoveMouse(MX, MY);

//logical programming-------------------------------

if (fClick == true)

{

if (BTN == 1) //left btn

{

if ((FrameCount >= int.Parse(txtCT.Text)) && (FrameCount < int.Parse(txtDCT.Text))) //single click

{

if (chkMouse.Checked){ MouseLeftKeyDown(); MouseLeftKeyUP();}

txtSysMsg.Text = "Left: Click" + Environment.NewLine + txtSysMsg.Text;

}

if ((FrameCount >= int.Parse(txtDCT.Text)) && (FrameCount < int.Parse(txtDT.Text))) //double click

{

if (chkMouse.Checked){MouseLeftKeyDown(); MouseLeftKeyUP();

MouseLeftKeyDown(); MouseLeftKeyUP();}

txtSysMsg.Text = "Left: Double Click" + Environment.NewLine + txtSysMsg.Text;

}

if (FrameCount >= int.Parse(txtDT.Text)) //dragging...

{

if (chkMouse.Checked){MouseLeftKeyDown();}

txtSysMsg.Text = "Left: Dragging..." + Environment.NewLine + txtSysMsg.Text;

fDrag = true;

}

}

if (BTN == 2) //right btn

{

if ((FrameCount >= int.Parse(txtCT.Text)) && (FrameCount < int.Parse(txtDCT.Text))) //single click

{

if (chkMouse.Checked){MouseRightKeyDown(); MouseRightKeyUP();}

txtSysMsg.Text = "Right: Click" + Environment.NewLine + txtSysMsg.Text;

}

if ((FrameCount >= int.Parse(txtDCT.Text)) && (FrameCount < int.Parse(txtDT.Text))) //double click

{

if (chkMouse.Checked){MouseRightKeyDown(); MouseRightKeyUP();

MouseRightKeyDown(); MouseRightKeyUP();}

txtSysMsg.Text = "Right: Double Click" + Environment.NewLine + txtSysMsg.Text;

}

}

}

fClick = false; //reset the flag

return;

}

if (FingerCount == 2)

{

if (fDrag == true)

{

MouseLeftKeyUP();

fDrag = false;

return;

}

if (fClick == false)

{

if (Y[0] < Y[1]) //geting the Middle btn

{

MX = X[0]; MY = Y[0];

if (X[1] < MX) //geting the Left OR Right btn

{

LX = X[1]; LY = Y[1];

BTN = 1; //left

}

else

{

RX = X[1]; RY = Y[1];

BTN = 2; //right

}

}

else

{

MX = X[1]; MY = Y[1];

//MX = X[0] - Frame; MY = Y[0] - Frame;//adjusting the frame

if (X[0] < MX) //geting the Left OR Right btn

{

LX = X[0]; LY = Y[0];

BTN = 1; //left

}

else

{

RX = X[0]; RY = Y[0];

BTN = 2; //right

}

}

fClick = true; FrameCount = 0; //start counting frames

}

else

{

FrameCount++; //increament frame count

txtSysMsg.Text = "" + FrameCount;

}

}

}

//-----------------------------------------------------------------------------------------

public void MoveMouse(int X,int Y)

{

lblX.Text = "" + X;

lblY.Text = "" + Y;

lblCMD.Text = ".";

if ((X > 106) && (X < 212) && (Y > 0) && (Y < 53))

{ //up

lblCMD.Text = "UP";

if (chkMouse.Checked) Cursor.Position = new Point(Cursor.Position.X, Cursor.Position.Y - int.Parse(txtSpeed.Text));

}

if ((X > 106) && (X < 212) && (Y > 106) && (Y < 160))

{ //down

lblCMD.Text = "DN";

if (chkMouse.Checked) Cursor.Position = new Point(Cursor.Position.X, Cursor.Position.Y + int.Parse(txtSpeed.Text));

}

if ((X > 212) && (X < 320) && (Y > 53) && (Y < 106))

{ //right

lblCMD.Text = "RT";

if (chkMouse.Checked) Cursor.Position = new Point(Cursor.Position.X + int.Parse(txtSpeed.Text), Cursor.Position.Y);

}

if ((X > 0) && (X < 106) && (Y > 53) && (Y < 106))

{ //left

lblCMD.Text = "LT";

if (chkMouse.Checked) Cursor.Position = new Point(Cursor.Position.X - int.Parse(txtSpeed.Text), Cursor.Position.Y);

}

if ((X > 212) && (X < 320) && (Y > 0) && (Y < 53))

{ //up-right

lblCMD.Text = "UR";

if (chkMouse.Checked) Cursor.Position = new Point(Cursor.Position.X + int.Parse(txtSpeed.Text), Cursor.Position.Y - int.Parse(txtSpeed.Text));

}

if ((X > 212) && (X < 320) && (Y > 106) && (Y < 160))

{ //down-right

lblCMD.Text = "DR";

if (chkMouse.Checked) Cursor.Position = new Point(Cursor.Position.X + int.Parse(txtSpeed.Text), Cursor.Position.Y + int.Parse(txtSpeed.Text));

}

if ((X > 0) && (X < 106) && (Y > 0) && (Y < 53))

{ //up-left

lblCMD.Text = "UL";

if (chkMouse.Checked) Cursor.Position = new Point(Cursor.Position.X - int.Parse(txtSpeed.Text), Cursor.Position.Y - int.Parse(txtSpeed.Text));

}

if ((X > 0) && (X < 106) && (Y > 106) && (Y < 160))

{ //down-left

lblCMD.Text = "DL";

if (chkMouse.Checked) Cursor.Position = new Point(Cursor.Position.X - int.Parse(txtSpeed.Text), Cursor.Position.Y + int.Parse(txtSpeed.Text));

}

}

//-----------------------------------------------------------------------------------------

//-----------------------------------------------------------------------------------------

public void DevelopGride()

{

LineSegment2D line = new LineSegment2D(new Point(106, 0), new Point(106,160));

imgMain.Draw(line, new Bgr(Color.YellowGreen), 2);

line = new LineSegment2D(new Point(212, 0), new Point(212, 160));

imgMain.Draw(line, new Bgr(Color.YellowGreen), 2);

line = new LineSegment2D(new Point(0, 53), new Point(320, 53));

imgMain.Draw(line, new Bgr(Color.YellowGreen), 2);

line = new LineSegment2D(new Point(0, 106), new Point(320, 106));

imgMain.Draw(line, new Bgr(Color.YellowGreen), 2);

line = new LineSegment2D(new Point(0, 160), new Point(320, 160));

imgMain.Draw(line, new Bgr(Color.YellowGreen), 2);

}

//-----------------------------------------------------------------------------------------

public void Extract\_Contour\_Hull\_Defects(Image<Gray, byte> BinaryHandImage)

{

#region variable Initialisation

Contour<Point> contours;

Contour<Point> biggestContour = null;

double Result1 = 0;

double Result2 = 0;

Seq<Point> hull;

Seq<MCvConvexityDefect> defects;

MCvConvexityDefect[] defectArray;

MemStorage storage;

#endregion

contours = BinaryHandImage.FindContours(Emgu.CV.CvEnum.CHAIN\_APPROX\_METHOD.CV\_CHAIN\_APPROX\_SIMPLE, Emgu.CV.CvEnum.RETR\_TYPE.CV\_RETR\_LIST);

while (contours != null)

{

Result1 = contours.Area;

if (Result1 > Result2)

{

Result2 = Result1;

biggestContour = contours;

}

contours = contours.HNext;

}

if (biggestContour != null)

{

biggestContour = biggestContour.ApproxPoly(0.00000001, 0, new MemStorage());

imgMain.Draw(biggestContour, new Bgr(Color.Black), 2);

// find the palm hand area using convexityDefect

hull = biggestContour.GetConvexHull(Emgu.CV.CvEnum.ORIENTATION.CV\_CLOCKWISE);

imgMain.DrawPolyline(hull.ToArray(), true, new Bgr(Color.White), 2);

// find defect area

storage = new MemStorage();

defects = biggestContour.GetConvexityDefacts(storage, Emgu.CV.CvEnum.ORIENTATION.CV\_CLOCKWISE);

defectArray = defects.ToArray();

#region calculate distane between every depth point and its end Point

int max = 0;

int[] distance = new int[defects.Total];

for (int i = 0; i < defects.Total; i++)

{

distance[i] = (int)Math.Sqrt(Math.Pow(defectArray[i].DepthPoint.X - defectArray[i].EndPoint.X, 2) + Math.Pow(defectArray[i].DepthPoint.Y - defectArray[i].EndPoint.Y, 2));

max = (int)Math.Max(max, distance[i]);

}

#endregion

#region find depth point that is base of the finger and assign it to importantDepthPoint

Contour<Point> importantDepthPoint = new Contour<Point>(new MemStorage());

Contour<Point> importantEndPoint = new Contour<Point>(new MemStorage());

int num = 0;

for (int i = 0; i < defects.Total; i++)

{

if (distance[i] > MinimumDistanceDepthPointToEndPointRatio \* max)

{

importantDepthPoint.Insert(num, new Point(defectArray[i].DepthPoint.X, defectArray[i].DepthPoint.Y));

importantEndPoint.Insert(num, new Point(defectArray[i].EndPoint.X, defectArray[i].EndPoint.Y));

//imgMain.Draw(new CircleF(defectArray[i].DepthPoint, 2), new Bgr(Color.Red), 2);

//imgMain.Draw(new CircleF(defectArray[i].EndPoint, 2), new Bgr(Color.GreenYellow), 2);

num++;

}

}

#endregion

PalmPointsCollection = new PointF[importantDepthPoint.Total];

Point[] importantDepthPointArray = importantDepthPoint.ToArray();

for (int i = 0; i < importantDepthPoint.Total; i++)

{

PalmPointsCollection[i] = new PointF(importantDepthPointArray[i].X, importantDepthPointArray[i].Y);

}

FingerPointsCollection = new PointF[importantEndPoint.Total];

Point[] importantEndPointArray = importantEndPoint.ToArray();

for (int i = 0; i < importantEndPoint.Total; i++)

{

FingerPointsCollection[i] = new PointF(importantEndPointArray[i].X, importantEndPointArray[i].Y);

}

}

}

//-----------------------------------------------------------------------------------------

private void Draw\_Gesture\_Features()

{

#region drawing PalmBoundingCircle

try

{

// find bounding rec for PalmPointsCollection

PalmBoundingCircle = PointCollection.MinEnclosingCircle(PalmPointsCollection);

}

catch

{

return;

}

// we treat center of the circle as the center of the palm

imgMain.Draw(PalmBoundingCircle, new Bgr(Color.Violet), 2);

imgMain.Draw(new CircleF(new PointF(PalmBoundingCircle.Center.X, PalmBoundingCircle.Center.Y), 2), new Bgr(Color.Violet), 5);

#endregion

#region drawing FingerBoundingBox

// find bounding rec for FingerPointsCollection

MCvBox2D box = PointCollection.MinAreaRect(FingerPointsCollection);

FingerBoundingBox = box.MinAreaRect();

// we treat center of the circle as the center of the palm

imgMain.Draw(FingerBoundingBox, new Bgr(Color.Cyan), 2);

imgMain.Draw(new CircleF(new PointF(box.center.X, box.center.Y), 2), new Bgr(Color.Cyan), 5);

#endregion

#region drawing all finger points

for (int i = 0; i < FingerPointsCollection.Count(); i++)

{

imgMain.Draw(new CircleF(FingerPointsCollection[i], 2), new Bgr(Color.Yellow), 5);

}

#endregion

#region drawing all palm points

for (int i = 0; i < PalmPointsCollection.Count(); i++)

{

imgMain.Draw(new CircleF(PalmPointsCollection[i], 2), new Bgr(Color.Red), 5);

}

#endregion

//#region drawing lines from fingers to center

//for (int i = 0; i < FingerPointsCollection.Count(); i++)

//{

// imgMain.Draw(new LineSegment2DF(FingerPointsCollection[i], PalmBoundingCircle.Center), new Bgr(Color.Yellow), 1);

//}

//#endregion

//#region drawing lines from palm to center

//for (int i = 0; i < PalmPointsCollection.Count(); i++)

//{

// imgMain.Draw(new LineSegment2DF(PalmPointsCollection[i], PalmBoundingCircle.Center), new Bgr(Color.Red), 1);

//}

//#endregion

//#region Filtered drawing lines from fingers to center

//for (int i = 0; i < FingerPointsCollection.Count(); i++)

//{

// if (FingerPointsCollection[i].Y < PalmBoundingCircle.Center.Y)

// {

// imgMain.Draw(new LineSegment2DF(FingerPointsCollection[i], PalmBoundingCircle.Center), new Bgr(Color.Yellow), 1);

// }

//}

//#endregion

//#region Filtered drawing lines from palm to center

//for (int i = 0; i < PalmPointsCollection.Count(); i++)

//{

// if (PalmPointsCollection[i].Y < PalmBoundingCircle.Center.Y)

// {

// imgMain.Draw(new LineSegment2DF(PalmPointsCollection[i], PalmBoundingCircle.Center), new Bgr(Color.Red), 1);

// }

//}

//#endregion

}

//\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\

//Skin Color filtering/////////////////////////////////////////////////////////////////////

//-----------------------------------------------------------------------------------------

public Image<Gray, byte> DetectSkinYCC(Image<Bgr, byte> Img, Ycc YCC\_min, Ycc YCC\_max)

{

Image<Ycc, Byte> currentYCrCbFrame = Img.Convert<Ycc, Byte>();

Image<Gray, byte> skin = new Image<Gray, byte>(Img.Width, Img.Height);

skin = currentYCrCbFrame.InRange(YCC\_min, YCC\_max);

return skin;

}

//-----------------------------------------------------------------------------------------

//-----------------------------------------------------------------------------------------

public Image<Gray, byte> DetectSkinHSV(Image<Bgr, byte> Img, Hsv HSV\_min, Hsv HSV\_max)

{

Image<Hsv, Byte> currentHSVFrame = Img.Convert<Hsv, Byte>();

Image<Gray, byte> skin = new Image<Gray, byte>(Img.Width, Img.Height);

skin = currentHSVFrame.InRange(HSV\_min, HSV\_max);

return skin;

}

//\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\

//Mouse CTRL //////////////////////////////////////////////////////////////////////////////

//-----------------------------------------------------------------------------------------

public void MouseLeftKeyDown()

{

mouse\_event(0x0002, 0, 0, 0, new IntPtr());

}

//-----------------------------------------------------------------------------------------

public void MouseLeftKeyUP()

{

mouse\_event(0x0004, 0, 0, 0, new IntPtr());

}

//-----------------------------------------------------------------------------------------

public void MouseRightKeyDown()

{

mouse\_event(0x0008, 0, 0, 0, new IntPtr());

}

//-----------------------------------------------------------------------------------------

public void MouseRightKeyUP()

{

mouse\_event(0x0010, 0, 0, 0, new IntPtr());

}

//-----------------------------------------------------------------------------------------

//\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\

}

}