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;

using Emgu.Util;

using Emgu.CV.Structure;

using System.IO.Ports;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

using System.Security.Cryptography;

namespace TestCam

{

public partial class Form1 : Form

{

static SerialPort \_serialPort;

public byte[] Buff = new byte[2];

private double Px2CmScale;

private Capture capture;

private Image<Bgr, Byte> IMG;

private Image<Gray, Byte> R\_frame;

private Image<Gray, Byte> G\_frame;

private Image<Gray, Byte> B\_frame;

private Image<Gray, Byte> GrayImg;

private Image<Gray, Byte> R\_Img\_seg;

private Image<Gray, Byte> R\_Img\_cor;

private Image<Gray, Byte> B\_Img\_seg;

private Image<Gray, Byte> B\_Img\_cor;

public double M1 = 90;

public double M1error = -3.5;

public double M2 = 90;

public double M2error = -13;

public double Th1;

public double Th2;

public int redNo;

public Form1()

{

\_serialPort = new SerialPort();

\_serialPort.PortName = "COM4";

\_serialPort.BaudRate = 9600;

\_serialPort.Open();

InitializeComponent();

textBox7.Text = M1error.ToString();

textBox3.Text = M2error.ToString();

}

private void processFrame(object sender, EventArgs e)

{

if (capture == null)

{

try

{

capture = new Capture();

}

catch (NullReferenceException excpt)

{

MessageBox.Show(excpt.Message);

}

}

IMG = capture.QueryFrame();

R\_frame = IMG[2].Copy();

G\_frame = IMG[1].Copy();

B\_frame = IMG[0].Copy();

GrayImg = IMG.Convert<Gray, Byte>();

R\_Img\_seg = IMG.Convert<Gray, Byte>();

R\_Img\_cor = IMG.Convert<Gray, Byte>();

B\_Img\_seg = IMG.Convert<Gray, Byte>();

B\_Img\_cor = IMG.Convert<Gray, Byte>();

label14.Text = "Width: " + GrayImg.Width.ToString();

label15.Text = "Height: " + GrayImg.Height.ToString();

int r\_th, b\_th, r\_cor, b\_cor;

r\_th = trackBar1.Value;

b\_th = trackBar2.Value;

r\_cor = trackBar3.Value;

b\_cor = trackBar4.Value;

for (int i = 0; i < GrayImg.Width; i++)

{

for (int j = 0; j < GrayImg.Height; j++)

{

if ((R\_frame[j, i].Intensity >= r\_th) && (B\_frame[j, i].Intensity + G\_frame[j, i].Intensity) < r\_th)

R\_Img\_seg.Data[j, i, 0] = 255;

else

R\_Img\_seg.Data[j, i, 0] = 0;

}

}

R\_Img\_cor = R\_Img\_seg.Copy();

for (int count = 0; count < r\_cor; count++)

{

for (int i = 1; i < GrayImg.Width - 1; i++)

for (int j = 1; j < GrayImg.Height - 1; j++)

if (R\_Img\_seg[j, i].Intensity != 0)

{

if ((R\_Img\_seg[j, i + 1].Intensity == 0) ||

(R\_Img\_seg[j - 1, i - 1].Intensity == 0) ||

(R\_Img\_seg[j - 1, i].Intensity == 0) ||

(R\_Img\_seg[j - 1, i + 1].Intensity == 0) ||

(R\_Img\_seg[j + 1, i + 1].Intensity == 0) ||

(R\_Img\_seg[j + 1, i].Intensity == 0) ||

(R\_Img\_seg[j + 1, i - 1].Intensity == 0))

R\_Img\_cor.Data[j, i, 0] = 0;

else R\_Img\_cor.Data[j, i, 0] = 255;

}

else

R\_Img\_cor.Data[j, i, 0] = 0;

R\_Img\_cor.CopyTo(R\_Img\_seg);

}

for (int i = 0; i < GrayImg.Width; i++)

{

for (int j = 0; j < GrayImg.Height; j++)

{

if (((B\_frame[j, i].Intensity < b\_th) && (R\_frame[j, i].Intensity) < b\_th) && (G\_frame[j, i].Intensity) < b\_th)

B\_Img\_seg.Data[j, i, 0] = 255;

else

B\_Img\_seg.Data[j, i, 0] = 0;

}

}

B\_Img\_cor = B\_Img\_seg.Copy();

for (int count = 0; count < b\_cor; count++)

{

for (int i = 1; i < GrayImg.Width - 1; i++)

for (int j = 1; j < GrayImg.Height - 1; j++)

if (B\_Img\_seg[j, i].Intensity != 0)

{

if ((B\_Img\_seg[j, i + 1].Intensity == 0) ||

(B\_Img\_seg[j - 1, i - 1].Intensity == 0) ||

(B\_Img\_seg[j - 1, i].Intensity == 0) ||

(B\_Img\_seg[j - 1, i + 1].Intensity == 0) ||

(B\_Img\_seg[j + 1, i + 1].Intensity == 0) ||

(B\_Img\_seg[j + 1, i].Intensity == 0) ||

(B\_Img\_seg[j + 1, i - 1].Intensity == 0))

B\_Img\_cor.Data[j, i, 0] = 0;

else B\_Img\_cor.Data[j, i, 0] = 255;

}

else

B\_Img\_cor.Data[j, i, 0] = 0;

B\_Img\_cor.CopyTo(B\_Img\_seg);

}

try

{

imageBox1.Image = IMG;

imageBox2.Image = R\_Img\_cor;

imageBox3.Image = B\_Img\_cor;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void button1\_Click(object sender, EventArgs e)

{

Application.Idle += processFrame;

button1.Enabled = false;

button2.Enabled = true;

}

private void button2\_Click(object sender, EventArgs e)

{

Application.Idle -= processFrame;

button1.Enabled = true;

button2.Enabled = false;

}

private void button3\_Click(object sender, EventArgs e)

{

IMG.Save("Image" + ".jpg");

}

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

{

int Xpx = 0;

int Ypx = 0;

int n = 0;

for (int i = 0; i < GrayImg.Width; i++)

for (int j = 0; j < GrayImg.Height; j++)

{

if (B\_Img\_cor[j, i].Intensity > 128)

{

Xpx += i;

Ypx += j;

n++;

}

}

if (n>0) {

Xpx = Xpx / n;

Ypx = Ypx / n;

label8.Text = "X px: " + Xpx.ToString();

label13.Text = "Y px: " + Ypx.ToString();

double Py = Xpx-(GrayImg.Width/2);

double Pz = -(Ypx - (GrayImg.Height / 2));

double Xcm;

double.TryParse(textBox1.Text, out Xcm);

double Ycm = Py \* Px2CmScale;

double Zcm = Pz \* Px2CmScale + 25;

textBox2.Text = Ycm.ToString("0.00");

textBox4.Text = Zcm.ToString("0.00");

Th1 = Math.Atan(Ycm / Xcm);

Th2 = Math.Atan(((Zcm) / Ycm) \* Math.Sin(Th1)) \* (180 / Math.PI);

Th1 = Th1 \* (180 / Math.PI);

textBox6.Text = Th1.ToString("0.00");

textBox5.Text = Th2.ToString("0.00");

double.TryParse(textBox7.Text.ToString(), out M1error);

double.TryParse(textBox3.Text.ToString(), out M2error);

Th1 = (int)((90 - M1error) - Th1);

Th2 = (int)((90 - M2error) - Th2);

Buff[0] = (byte)Th1;

Buff[1] = (byte)Th2;

\_serialPort.Write(Buff, 0, 2);

}

}

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

{

double[] proj = new double[GrayImg.Width];

for (int i = 0; i < GrayImg.Width; i++)

{

double column = 0;

for (int j = 0; j < GrayImg.Height; j++)

{

proj[i] = column = column + ((R\_Img\_cor[j, i].Intensity) / 255);

}

}

int k = 0;

double sum = 0;

while (k < proj.Length && proj[k] == 0) { k++; }

k += 5;

int start = k;

for (int i = 0; i < 2; i++)

{

while (k < proj.Length && proj[k] != 0) k++;

k += 5;

while (k < (GrayImg.Width - 5) && proj[k] == 0) k++;

k += 5;

int end = k;

sum = sum + (end - start);

start = end;

}

int.TryParse(textBox8.Text.ToString(), out redNo);

double Avg = sum / (redNo-1);

Px2CmScale = 20.0 / Avg;

label16.Text = "Scale: " + Px2CmScale.ToString("0.00");

label12.Text = "Average: " + Avg.ToString();

}

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

{

Th1 = 95;

Th2 = 95;

Buff[0] = (byte)Th1;

Buff[1] = (byte)Th2;

\_serialPort.Write(Buff, 0, 2);

}

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

{

Th1 = 45;

Th2 = 45;

Buff[0] = (byte)Th1;

Buff[1] = (byte)Th2;

\_serialPort.Write(Buff, 0, 2);

}

}

}