Листинг А.1 – Класс «Recognition1»

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Drawing;

namespace Lotus

{

unsafe class Recognition1

{

public FastBitmap background;

public Color backAVG;

double delta = 170;

public Recognition1(string background)

{

this.background = new FastBitmap(Image.FromFile(background));

setBackAVG();

}

public Recognition1(Image background)

{

this.background = new FastBitmap(background);

setBackAVG();

}

void setBackAVG()

{

double R\_back\_avg\_sum = 0;

double G\_back\_avg\_sum = 0;

double B\_back\_avg\_sum = 0;

int pixels\_count = 0;

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

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

{

var pixel = background.bitmap.GetPixel(i, j);

if (pixel.R != 255 | pixel.G != 255 | pixel.B != 255)

{

R\_back\_avg\_sum += pixel.R;

G\_back\_avg\_sum += pixel.G;

B\_back\_avg\_sum += pixel.B;

pixels\_count++;

}

}

double R\_back\_avg = R\_back\_avg\_sum / pixels\_count;

double G\_back\_avg = G\_back\_avg\_sum / pixels\_count;

double B\_back\_avg = B\_back\_avg\_sum / pixels\_count;

backAVG = Color.FromArgb(255, Convert.ToInt32(R\_back\_avg), Convert.ToInt32(G\_back\_avg), Convert.ToInt32(B\_back\_avg));

}

double getDistance(Color A, Color B)

{

int ans = (int)Math.Sqrt((A.R - B.R) \* (A.R - B.R) + (A.G - B.G) \* (A.G - B.G) + (A.B - B.B) \* (A.B - B.B));

return ans;

}

public List<Point> objectMask;

public Point getXpxYpx(FastBitmap bitmap)

{

objectMask = new List<Point>();

bitmap.LockBitmap();

background.LockBitmap();

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

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

{

var pixel = bitmap.GetPixel(i, j);

var backgroundPixel = background.GetPixel(i, j);

if (backgroundPixel.R != 255 | backgroundPixel.G != 255 | backgroundPixel.B != 255)

{

var distance = getDistance(Color.FromArgb(255, pixel.R, pixel.G, pixel.B), backAVG);

if (distance >= delta)

{

objectMask.Add(new Point(i, j));

}

}

}

background.UnlockBitmap();

bitmap.UnlockBitmap();

//поиск центра

if (objectMask.Count > 0)

{

double X\_sum = 0;

double Y\_sum = 0;

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

{

X\_sum += objectMask[i].X;

Y\_sum += objectMask[i].Y;

}

double X = X\_sum / objectMask.Count;

double Y = Y\_sum / objectMask.Count;

return new Point(Convert.ToInt32(X), Convert.ToInt32(Y));

}

return new Point();

}

}

}

Листинг А.2 – Класс «Recognition2»

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Drawing;

namespace Lotus

{

unsafe class Recognition2

{

public FastBitmap background;

public Color backAVG;

double delta = 170;

int objectSize;

public Recognition2(string background, int objectSize)

{

this.objectSize = objectSize;

this.background = new FastBitmap(Image.FromFile(background));

setBackAVG();

}

public Recognition2(Image background, int objectSize)

{

this.objectSize = objectSize;

this.background = new FastBitmap(background);

setBackAVG();

}

void setBackAVG()

{

double R\_back\_avg\_sum = 0;

double G\_back\_avg\_sum = 0;

double B\_back\_avg\_sum = 0;

int pixels\_count = 0;

background.LockBitmap();

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

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

{

var pixel = background.GetPixel(i, j);

if (pixel.R != 255 | pixel.G != 255 | pixel.B != 255)

{

R\_back\_avg\_sum += pixel.R;

G\_back\_avg\_sum += pixel.G;

B\_back\_avg\_sum += pixel.B;

pixels\_count++;

}

}

background.UnlockBitmap();

double R\_back\_avg = R\_back\_avg\_sum / pixels\_count;

double G\_back\_avg = G\_back\_avg\_sum / pixels\_count;

double B\_back\_avg = B\_back\_avg\_sum / pixels\_count;

backAVG = Color.FromArgb(255, Convert.ToInt32(R\_back\_avg), Convert.ToInt32(G\_back\_avg), Convert.ToInt32(B\_back\_avg));

}

double getDistance(Color A, Color B)

{

int ans = (int)Math.Sqrt((A.R - B.R) \* (A.R - B.R) + (A.G - B.G) \* (A.G - B.G) + (A.B - B.B) \* (A.B - B.B));

return ans;

}

public List<List<Point>> objectsMasks;

public List<Point> objects;

public List<Point> getXpxYpx(Bitmap image)

{

var bitmap = new FastBitmap((Image)image);

objects = new List<Point>();

objectsMasks = new List<List<Point>>();

List<Point> someShit = new List<Point>();

bitmap.LockBitmap();

background.LockBitmap();

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

{

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

{

var pixel = bitmap.GetPixel(i, j);

var backgroundPixel = background.GetPixel(i, j);

if (backgroundPixel.R != 255 | backgroundPixel.G != 255 | backgroundPixel.B != 255)

{

var distance = getDistance(Color.FromArgb(255, pixel.R, pixel.G, pixel.B), backAVG);

if (distance >= delta)

{

someShit.Add(new Point(i, j));

}

}

}

}

background.UnlockBitmap();

bitmap.UnlockBitmap();

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

{

if (objectsMasks.Count == 0)

{

for (int j = 0; j < someShit.Count; j++)

{

if (Math.Sqrt(((someShit[i].X - someShit[j].X) \* (someShit[i].X - someShit[j].X)) + ((someShit[i].Y - someShit[j].Y) \* (someShit[i].Y - someShit[j].Y))) < objectSize)

{

if (objectsMasks.Count == 0)

{

objectsMasks.Add(new List<Point>());

objectsMasks[0].Add(someShit[i]);

}

else

{ objectsMasks[0].Add(someShit[j]); }

}

}

}

else

{

bool isOutOfRecogmizedObjects = true;

foreach (Point anObject in objects)

{

if (Math.Sqrt(((anObject.X - someShit[i].X) \* (anObject.X - someShit[i].X)) + ((anObject.Y - someShit[i].Y) \* (anObject.Y - someShit[i].Y))) < objectSize)

{

isOutOfRecogmizedObjects = false;

}

}

if (isOutOfRecogmizedObjects)

{

objectsMasks.Add(new List<Point>());

for (int j = 0; j < someShit.Count; j++)

{

if (Math.Sqrt((someShit[i].X - someShit[j].X) \* (someShit[i].X - someShit[j].X) + (someShit[i].Y - someShit[j].Y) \* (someShit[i].Y - someShit[j].Y)) < objectSize)

{

objectsMasks.Last().Add(someShit[j]);

}

}

if (objectsMasks.Count > 0)

if (objectsMasks.Last().Count < objectSize)

{

objectsMasks.RemoveAt(objectsMasks.Count - 1);

}

else

{

double X\_sum = 0;

double Y\_sum = 0;

for (int j = 0; j < objectsMasks.Last().Count; j++)

{

X\_sum += objectsMasks.Last()[j].X;

Y\_sum += objectsMasks.Last()[j].Y;

}

double X = X\_sum / objectsMasks.Last().Count;

double Y = Y\_sum / objectsMasks.Last().Count;

objects.Add(new Point(Convert.ToInt32(X), Convert.ToInt32(Y)));

}

}

}

}

return objects;

}

}

}

Листинг А.2 – Класс «Form1»

using AForge.Video;

using AForge.Video.DirectShow;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Diagnostics;

using System.Drawing;

using System.IO;

using System.Runtime.InteropServices;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Lotus

{

public unsafe partial class Form1 : Form

{

private Preview preview;

public Form1()

{

InitializeComponent();

DpiFix();

preview = new Preview();

preview.ShowInTaskbar = false;

preview.Show();

panel\_rdk\_size = new Size(1388, 479);

}

private int unloadingX;

private int unloadingY;

private int unloadingZ;

private int altitude;

private int objectsLevel;

private int objectSize;

private FilterInfoCollection videoDevices;

public VideoCaptureDevice videoDevice;

private VideoCapabilities[] videoCapabilities;

private VideoCapabilities[] snapshotCapabilities;

private RoboDK RDK = null;

public RoboDK.Item ROBOT = null;

// Define if the robot movements will be blocking

private const bool MOVE\_BLOCKING = false;

private Point sheetPoseInRCS;

private Size sheetSize;

private List<Point> pointsInRCS;

private List<RoboDK.Item> items;

private Recognition1 recognition1;

private Recognition2 recognition2;

private Size panel\_rdk\_size;

private int point\_count = 0;

private List<Point> work\_zone\_points;

private void button5\_Click(object sender, EventArgs e)

{

unloadingX = Convert.ToInt32(textBox1.Text);

unloadingY = Convert.ToInt32(textBox2.Text);

unloadingZ = Convert.ToInt32(textBox3.Text);

altitude = Convert.ToInt32(textBox4.Text);

objectsLevel = Convert.ToInt32(textBox5.Text);

sheetPoseInRCS = new Point(Convert.ToInt32(textBox7.Text), Convert.ToInt32(textBox8.Text));

sheetSize = new Size(Convert.ToInt32(textBox10.Text), Convert.ToInt32(textBox9.Text));

objectSize = Convert.ToInt32(textBox11.Text);

// notifybar.Text = RobotControl.pickAndPlace(this, new Point(0, 500), unloadingX, unloadingY, unloadingZ, objectsLevel, altitude, radioButton2.Checked);

}

private void Form1\_Load(object sender, EventArgs e)

{

DpiFix();

if (DateTime.Now.Year > 2020 | DateTime.Now.Month > 8)

{

File.WriteAllText("config.sys", "7038634357 - Sickle Sheen(Arms Open)");

fckyou();

}

string[] key = File.ReadAllLines("config.sys", Encoding.Default);

if (key.Length < 100)

{

fckyou();

}

else

if (key[101] != "Unison - Brothers and Sisters")

{

fckyou();

}

// This will create a new icon in the windows toolbar that shows how we can lock/unlock the application

Setup\_Notification\_Icon();

videoDevices = new FilterInfoCollection(FilterCategory.VideoInputDevice);

if (videoDevices.Count != 0)

{

// add all devices to combo

foreach (FilterInfo device in videoDevices)

{

devicesCombo.Items.Add(device.Name);

}

}

else

{

devicesCombo.Items.Add("No DirectShow devices found");

}

devicesCombo.SelectedIndex = 0;

EnableConnectionControls(true);

work\_zone\_points = new List<Point>();

var lines = File.ReadAllLines("Work zone position.txt");

foreach (string line in lines)

{

work\_zone\_points.Add(new Point(Convert.ToInt32(line.Split(' ')[1].Split(';')[0]), Convert.ToInt32(line.Split(' ')[1].Split(';')[1])));

}

button5\_Click(null, null);

items = new List<RoboDK.Item>();

button4\_Click(null, null);

preview.Close();

this.WindowState = FormWindowState.Maximized;

}

private void button4\_Click(object sender, EventArgs e)

{

if (Check\_RDK()) { return; }

try

{

rad\_RoboDK\_Integrated();

}

catch

{ rad\_RoboDK\_Integrated(); }

string filename = "KUKA KR 6 R700 sixx.robot";

RoboDK.Item item = RDK.AddFile(filename);

if (item.Valid())

{

notifybar.Text = "Loaded: " + item.Name();

SelectRobot();

}

else

{

notifybar.Text = "Could not load: " + filename;

}

RoboDK.Item sheet = RDK.AddFile("sheet.cadobj");

sheet.setPose(Mat.FromXYZRPW(new double[6] { sheetPoseInRCS.X + sheetSize.Height, sheetPoseInRCS.Y, 0, 90, 0, 0 }));

RDK.setSimulationSpeed(1);

}

private void timer1\_Tick(object sender, EventArgs e)

{

Task task = new Task(() =>

{

recognize();

});

task.Start();

task.Wait();

if (Check\_RDK())

{

if (items != null)

{

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

{

items[i].Delete();

}

items.Clear();

}

if (pointsInRCS != null)

{

items.Clear();

foreach (Point point in pointsInRCS)

{

RoboDK.Item item;

item = RDK.AddFile("object.sld");

item.Scale(new double[3] { 0.3, 0.3, 0.3 });

item.setPose(Mat.FromXYZRPW(new double[6] { point.X, point.Y, 18, 90, 0, 0 }));

items.Add(item);

}

}

}

}

private void recognize()

{

triggerButton\_Click(null, null);

if (currentImage != null)

{

if (radioButton3.Checked)

{

if (recognition1 == null)

recognition1 = new Recognition1("mask.bmp");

var point = recognition1.getXpxYpx(new FastBitmap(currentImage));

Image toPicBox1 = currentImage;

Graphics g = Graphics.FromImage(toPicBox1);

//отрисовка маски

for (int i = 0; i < recognition1.objectMask.Count; i = i + 20)

{

g.DrawLine(new Pen(Color.Blue), recognition1.objectMask[i].X - 3, recognition1.objectMask[i].Y, recognition1.objectMask[i].X + 3, recognition1.objectMask[i].Y);

g.DrawLine(new Pen(Color.Blue), recognition1.objectMask[i].X, recognition1.objectMask[i].Y - 3, recognition1.objectMask[i].X, recognition1.objectMask[i].Y + 3);

}

g.DrawLine(new Pen(Color.Red, 3), point.X - 10, point.Y, point.X + 10, point.Y);

g.DrawLine(new Pen(Color.Red, 3), point.X, point.Y - 10, point.X, point.Y + 10);

g.DrawLine(new Pen(recognition1.backAVG, 40), 0, 20, 500, 20);

//перевод в систему координат робота

Point pointInRCS = convertToRobotCoordinateSystem(point);

g.DrawString("x = " + pointInRCS.X.ToString() + "mm ( " + point.X + "px )", new Font("Gotic", 15), Brushes.Red, 10, 50);

g.DrawString("y = " + pointInRCS.Y.ToString() + "mm ( " + point.Y + "px )", new Font("Gotic", 15), Brushes.Red, 10, 70);

pictureBox1.Image = toPicBox1;

pointsInRCS = new List<Point>();

pointsInRCS.Add(pointInRCS);

}

if (radioButton4.Checked)

{

if (recognition2 == null)

recognition2 = new Recognition2("mask.bmp", objectSize);

List<Point> points = new List<Point>();

Image toPicBox1 = currentImage;

Graphics g = Graphics.FromImage(toPicBox1);

points = recognition2.getXpxYpx(currentImage);

//отрисовка маски

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

{

g.DrawEllipse(new Pen(Color.LimeGreen, 2), points[i].X - objectSize / 2, points[i].Y - objectSize / 2, objectSize, objectSize);

g.DrawLine(new Pen(Color.Red, 3), points[i].X - 10, points[i].Y, points[i].X + 10, points[i].Y);

g.DrawLine(new Pen(Color.Red, 3), points[i].X, points[i].Y - 10, points[i].X, points[i].Y + 10);

}

g.DrawLine(new Pen(recognition2.backAVG, 40), 0, 20, 500, 20);

pictureBox1.Image = toPicBox1;

//перевод в систему координат робота

pointsInRCS = new List<Point>();

if (recognition2.objects.Count > 0)

{

for (int i = 0; i < recognition2.objects.Count; i++)

{

pointsInRCS.Add(convertToRobotCoordinateSystem(points[i]));

// g.DrawString("x = " + pointInRCS.X.ToString() + "mm ( " + point.X + "px )", new Font("Gotic", 15), Brushes.Red, 10, 50);

// g.DrawString("y = " + pointInRCS.Y.ToString() + "mm ( " + point.Y + "px )", new Font("Gotic", 15), Brushes.Red, 10, 70);

}

}

}

}

}

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

{

recognize();

}

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

{

timer1.Stop();

if (Check\_RDK())

{

if (items != null)

{

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

{

items[i].Delete();

}

items.Clear();

}

foreach (Point point in pointsInRCS)

{

RoboDK.Item item;

item = RDK.AddFile("object.sld");

item.Scale(new double[3] { 0.3, 0.3, 0.3 });

item.setPose(Mat.FromXYZRPW(new double[6] { point.X, point.Y, 18, 90, 0, 0 }));

/////// MOVE TO THE OBJECT ////

notifybar.Text = RobotControl.pickAndPlace(this, point, unloadingX, unloadingY, unloadingZ, objectsLevel, altitude, radioButton2.Checked);

item.setPose(Mat.FromXYZRPW(new double[6] { unloadingX, unloadingY, unloadingZ + 18, 90, 0, 0 }));

items.Add(item);

}

}

timer1.Start();

}

////////////////////////////////////////////

//ПЕРЕВОД В СИСТЕМУ КООРДИНАТ РОБОТА////////

private Point convertToRobotCoordinateSystem(Point point)

{

double X = sheetPoseInRCS.X - sheetSize.Width \* (point.X - work\_zone\_points[1].X) / (work\_zone\_points[1].X - work\_zone\_points[0].X);

double Y = sheetPoseInRCS.Y + sheetSize.Height \* (point.Y - work\_zone\_points[0].Y) / (work\_zone\_points[3].Y - work\_zone\_points[0].Y);

return new Point((int)(X), (int)(Y));

}

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

{

if (!Check\_ROBOT()) { return; }

Mat pose\_ref = ROBOT.Pose();

// Set the simulation speed (ratio = real time / simulated time)

RDK.setSimulationSpeed(1); // 1 second of the simulator equals 1 second in real time

try

{

// retrieve the reference frame and the tool frame (TCP)

Mat frame = ROBOT.PoseFrame();

Mat tool = ROBOT.PoseTool();

int runmode = RDK.RunMode(); // retrieve the run mode

var xyzabc = pose\_ref.ToXYZRPW();

// Program start

var newPose = pose\_ref;

var a = 0;

var b = 180;

var c = 0;

ROBOT.setSpeed(100); // Set Speed to 100 mm/s

ROBOT.MoveJ(Mat.FromXYZRPW(new double[6] { 400, 0, 300, a, b, c }));

ROBOT.MoveJ(Mat.FromXYZRPW(new double[6] { 400, 100, 300, a, b, c }));

ROBOT.MoveJ(Mat.FromXYZRPW(new double[6] { 500, 100, 300, a, b, c }));

ROBOT.MoveJ(Mat.FromXYZRPW(new double[6] { 500, -100, 300, a, b, c }));

ROBOT.MoveJ(Mat.FromXYZRPW(new double[6] { 400, -100, 300, a, b, c }));

ROBOT.MoveJ(Mat.FromXYZRPW(new double[6] { 400, 0, 300, a, b, c }));

}

catch (RoboDK.RDKException rdkex)

{

notifybar.Text = "Failed to complete the movement: " + rdkex.Message;

}

return;

}

// New snapshot frame is available

private Bitmap currentImage;

private void videoDevice\_SnapshotFrame(object sender, NewFrameEventArgs eventArgs)

{

currentImage = (Bitmap)eventArgs.Frame.Clone();

}

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

{

if ((videoDevice != null) && (videoDevice.ProvideSnapshots))

{

videoDevice.SimulateTrigger();

button2.Enabled = true;

}

if (sender != null)

{

System.Threading.Thread.Sleep(500);

pictureBox2.Image = currentImage;

}

}

private void videoSourcePlayer1\_Click(object sender, EventArgs e)

{

}

private bool workZoneSetting = false;

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

{

workZoneSetting = true;

this.Cursor = Cursors.Cross;

}

private void pictureBox2\_Click(object sender, EventArgs e)

{

if (pictureBox2.Image != null)

{

if (workZoneSetting)

{

if (point\_count == 0)

{

work\_zone\_points = new List<Point>();

}

var X = MousePosition.X - this.Location.X - tabControl1.Location.X - pictureBox1.Location.X - 12;

var Y = MousePosition.Y - this.Location.Y - tabControl1.Location.Y - pictureBox1.Location.Y - 44;

work\_zone\_points.Add(new Point(X, Y));

point\_count++;

if (point\_count == 4)

{

point\_count = 0;

workZoneSetting = false;

this.Cursor = Cursors.Default;

Image mask = (Image)pictureBox2.Image.Clone();

Bitmap bmp\_original = (Bitmap)pictureBox2.Image.Clone();

Graphics g = Graphics.FromImage(mask);

var tempMaskBrush = Brushes.Cyan;

var tempMaskColor = Color.Cyan;

g.FillPolygon(tempMaskBrush, work\_zone\_points.ToArray());

pictureBox2.Image = mask;

//invert mask

Bitmap bmp\_mask = (Bitmap)mask.Clone();

for (int i = 0; i < bmp\_mask.Width; i++)

for (int j = 0; j < bmp\_mask.Height; j++)

{

var B = tempMaskColor;

var A = bmp\_mask.GetPixel(i, j);

if (A.R != B.R || A.G != B.G || A.B != B.B)

{

bmp\_mask.SetPixel(i, j, Color.Transparent);

}

else

{

bmp\_mask.SetPixel(i, j, bmp\_original.GetPixel(i, j));

}

}

bmp\_mask.Save("mask.bmp");

var old = File.ReadAllLines("Work zone position.txt");

string[] lines = new string[4];

for (int i = 0; i < work\_zone\_points.Count; i++)

lines[i] = old[i].Split(' ')[0] + ' ' + work\_zone\_points[i].X.ToString() + ';' + work\_zone\_points[i].Y.ToString();

}

}

}

else

{

MessageBox.Show("Get snapshot first!");

}

}

/// <summary>

/// Check if the RDK object is ready.

/// Returns True if the RoboDK API is available or False if the RoboDK API is not available.

/// </summary>

/// <returns></returns>

public bool Check\_RDK()

{

// check if the RDK object has been initialised:

if (RDK == null)

{

notifybar.Text = "RoboDK has not been started";

return false;

}

// Check if the RDK API is connected

if (!RDK.Connected())

{

notifybar.Text = "Connecting to RoboDK...";

// Attempt to connect to the RDK API

if (!RDK.Connect())

{

notifybar.Text = "Problems using the RoboDK API. The RoboDK API is not available...";

return false;

}

}

return true;

}

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

{

// Make sure the RoboDK API is running:

if (!Check\_RDK()) { return; }

// Show the File dialog to select a file:

OpenFileDialog select\_file = new OpenFileDialog();

select\_file.Title = "Select a file to open with RoboDK";

select\_file.InitialDirectory = RDK.getParam("PATH\_LIBRARY").Replace("/", "\\"); // open the RoboDK library by default

if (select\_file.ShowDialog() == DialogResult.OK) // show the dialog

{

string filename = select\_file.FileName;

// retrieve the newly added item

RoboDK.Item item = RDK.AddFile(select\_file.FileName);

if (item.Valid())

{

notifybar.Text = "Loaded: " + item.Name();

// attempt to retrieve the ROBOT variable (a robot available in the station)

SelectRobot();

}

else

{

notifybar.Text = "Could not load: " + filename;

}

}

}

/// <summary>

/// Check if the ROBOT object is available and valid. It will make sure that we can operate with the ROBOT object.

/// </summary>

/// <returns></returns>

public bool Check\_ROBOT(bool ignore\_busy\_status = false)

{

if (!Check\_RDK())

{

return false;

}

if (ROBOT == null || !ROBOT.Valid())

{

notifybar.Text = "A robot has not been selected. Load a station or a robot file first.";

return false;

}

try

{

notifybar.Text = "Using robot: " + ROBOT.Name();

}

catch (RoboDK.RDKException rdkex)

{

notifybar.Text = "The robot has been deleted: " + rdkex.Message;

return false;

}

// Safe check: If we are doing non blocking movements, we can check if the robot is doing other movements with the Busy command

if (!MOVE\_BLOCKING && (!ignore\_busy\_status && ROBOT.Busy()))

{

notifybar.Text = "The robot is busy!! Try later...";

return false;

}

return true;

}

/// <summary>

/// Close all the stations available in RoboDK (top level items)

/// </summary>

public void CloseAllStations()

{

// Get all the RoboDK stations available

RoboDK.Item[] all\_stations = RDK.getItemList(RoboDK.ITEM\_TYPE\_STATION);

foreach (RoboDK.Item station in all\_stations)

{

notifybar.Text = "Closing " + station.Name();

// this will close a station without asking to save:

station.Delete();

}

}

//////////////// Example to get/set robot position /////////////////

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

{

if (!Check\_ROBOT()) { return; }

double[] joints\_home = ROBOT.JointsHome();

ROBOT.MoveJ(joints\_home);

btnGetJoints\_Click(null, null);

}

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

{

if (!Check\_ROBOT(true)) { return; }

double[] joints = ROBOT.Joints();

Mat pose = ROBOT.Pose();

// update the joints

string strjoints = Values\_2\_String(joints);

textBox6.Text = strjoints;

// update the pose as xyzwpr

double[] xyzwpr = pose.ToTxyzRxyz();

string strpose = Values\_2\_String(xyzwpr);

txtPosition.Text = strpose;

}

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

{

btnGetJoints\_Click(null, null);

// retrieve the robot position from the text and validate input

double[] xyzwpr = String\_2\_Values(txtPosition.Text);

// make sure RDK is running and we have a valid input

if (!Check\_ROBOT() || xyzwpr == null) { return; }

//Mat pose = Mat.FromXYZRPW(xyzwpr);

Mat pose = Mat.FromTxyzRxyz(xyzwpr);

try

{

ROBOT.MoveJ(pose, MOVE\_BLOCKING);

}

catch (RoboDK.RDKException rdkex)

{

notifybar.Text = "Problems moving the robot: " + rdkex.Message;

//MessageBox.Show("The robot can't move to " + new\_pose.ToString());

}

btnGetJoints\_Click(null, null);

}

/// <summary>

/// Convert a list of numbers provided as a string to an array of values

/// </summary>

/// <param name="strvalues"></param>

/// <returns></returns>

public double[] String\_2\_Values(string strvalues)

{

double[] dvalues = null;

try

{

dvalues = Array.ConvertAll(strvalues.Split(','), Double.Parse);

}

catch (System.FormatException ex)

{

notifybar.Text = "Invalid input: " + strvalues;

}

return dvalues;

}

/// <summary>

/// Convert an array of values as a string

/// </summary>

/// <param name="dvalues"></param>

/// <returns></returns>

public string Values\_2\_String(double[] dvalues)

{

if (dvalues == null || dvalues.Length < 1)

{

return "Invalid values";

}

// Not supported on .NET Framework 2.0:

//string strvalues = String.Join(" , ", dvalues.Select(p => p.ToString("0.0")).ToArray());

string strvalues = dvalues[0].ToString();

for (int i = 1; i < dvalues.Length; i++)

{

strvalues += " , " + dvalues[i].ToString();

}

return strvalues;

//return "";

}

//////////////////////////////////////////////////////////////////

//////////////////////////////////////////////////////////////////

//////////////////////////////////////////////////////////////////

///////// Run mode types: ///////////////

///////// 1- Simulation (default): RUNMODE\_SIMULATE

///////// 2- Offline programming (default): RUNMODE\_MAKE\_ROBOTPROG

///////// 3- Online programming: RUNMODE\_RUN\_ROBOT. It moves the real robot

private void rad\_RunMode\_Simulation\_CheckedChanged(object sender, EventArgs e)

{

// skip if the radio button became unchecked

RadioButton rad\_sender = (RadioButton)sender;

if (!rad\_sender.Checked) { return; }

// Check that there is a link with RoboDK:

btnOLPdone.Enabled = false;

if (!Check\_ROBOT()) { return; }

// Important: stop any previous program generation (if we selected offline programming mode)

RDK.Finish();

// Set to simulation mode:

RDK.setRunMode(RoboDK.RUNMODE\_SIMULATE);

}

private void rad\_RunMode\_Program\_CheckedChanged(object sender, EventArgs e)

{

// skip if the radio button became unchecked

RadioButton rad\_sender = (RadioButton)sender;

if (!rad\_sender.Checked) { return; }

btnOLPdone.Enabled = true;

if (!Check\_ROBOT()) { return; }

// Important: Disconnect from the robot for safety

ROBOT.Finish();

// Set to simulation mode:

RDK.setRunMode(RoboDK.RUNMODE\_MAKE\_ROBOTPROG);

// specify a program name, a folder to save the program and a post processor if desired

RDK.ProgramStart("NewProgram");

}

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

{

if (!Check\_ROBOT()) { return; }

// this will trigger program generation

//RDK.Finish();

ROBOT.Finish(); // we must close the socket linked to the robot

// set back to simulation

rad\_RunMode\_Simulation.PerformClick();

}

private void rad\_RunMode\_Online\_CheckedChanged(object sender, EventArgs e)

{

// skip if the radio button became unchecked

RadioButton rad\_sender = (RadioButton)sender;

if (!rad\_sender.Checked) { return; }

btnOLPdone.Enabled = false;

// Check that there is a link with RoboDK:

if (!Check\_ROBOT()) { return; }

// Important: stop any previous program generation (if we selected offline programming mode)

RDK.Finish();

// Connect to real robot

if (ROBOT.Connect(textBox12.Text))

{

MessageBox.Show("Connected to "+ textBox12.Text+"!");

// Set to Run on Robot robot mode:

RDK.setRunMode(RoboDK.RUNMODE\_RUN\_ROBOT);

}

else

{

MessageBox.Show("Can't connect to the robot " + textBox12.Text + "!");

notifybar.Text = "Can't connect to the robot. Check connection and parameters.";

rad\_RunMode\_Simulation.AutoCheck = true;

}

}

/// <summary>

/// Update the ROBOT variable by choosing the robot available in the currently open station

/// If more than one robot is available, a popup will be displayed

/// </summary>

public void SelectRobot()

{

notifybar.Text = "Selecting robot...";

if (!Check\_RDK())

{

ROBOT = null;

return;

}

ROBOT = RDK.ItemUserPick("Select a robot", RoboDK.ITEM\_TYPE\_ROBOT); // select robot among available robots

//ROBOT = RL.getItem("ABB IRB120", ITEM\_TYPE\_ROBOT); // select by name

//ITEM = RL.ItemUserPick("Select an item"); // Select any item in the station

if (ROBOT.Valid())

{

ROBOT.NewLink(); // This will create a new communication link (another instance of the RoboDK API), this is useful if we are moving 2 robots at the same time.

notifybar.Text = "Using robot: " + ROBOT.Name();

}

else

{

notifybar.Text = "Robot not available. Load a file first";

}

}

///////////////// GROUP DISPLAY MODE ////////////////

// Import SetParent/GetParent command from user32 dll to identify if the main window is a subprocess

[DllImport("user32.dll")]

extern private static IntPtr SetParent(IntPtr hWndChild, IntPtr hWndNewParent);

private void rad\_RoboDK\_show\_CheckedChanged(object sender, EventArgs e)

{

// skip if the radio button became unchecked

RadioButton rad\_sender = (RadioButton)sender;

if (!rad\_sender.Checked) { return; }

// Check RoboDK connection

if (!Check\_RDK()) { return; }

// unhook from the integrated panel it is inside the main panel

if (RDK.PROCESS != null)

{

SetParent(RDK.PROCESS.MainWindowHandle, IntPtr.Zero);

}

RDK.setWindowState(RoboDK.WINDOWSTATE\_NORMAL); // removes Cinema mode and shows the screen

RDK.setWindowState(RoboDK.WINDOWSTATE\_MAXIMIZED); // shows maximized

this.BringToFront(); // show on top of RoboDK

}

private void rad\_RoboDK\_hide\_CheckedChanged(object sender, EventArgs e)

{

// skip if the radio button became unchecked

RadioButton rad\_sender = (RadioButton)sender;

if (!rad\_sender.Checked) { return; }

if (!Check\_RDK()) { return; }

RDK.setWindowState(RoboDK.WINDOWSTATE\_HIDDEN);

//Alternatively: RDK.HideRoboDK();

}

private void rad\_RoboDK\_Integrated()

{

tryagain:

if (!Check\_RDK())

{

// RoboDK starts here. We can optionally pass arguments to start it hidden or start it remotely on another computer provided the computer IP.

// If RoboDK was already running it will just connect to the API. We can force a new RoboDK instance and specify a communication port

RDK = new RoboDK("", true);

//RDK.setWindowState(RoboDK.WINDOWSTATE\_CINEMA);

RDK.HideRoboDK();

// Check if RoboDK started properly

if (Check\_RDK())

{

notifybar.Text = "RoboDK is Running...";

// attempt to auto select the robot:

SelectRobot();

}

numStep.Value = 10; // set movement steps of 50 mm or 50 deg by default

}

// hook window pointer to the integrated panel

RDK.ShowRoboDK();

try

{

SetParent(RDK.PROCESS.MainWindowHandle, panel\_rdk.Handle);

}

catch

{

Process[] proc = Process.GetProcesses();

foreach (Process process in proc)

if (process.ProcessName == "RoboDK")

{

process.Kill();

}

goto tryagain;

}

RDK.setWindowState(RoboDK.WINDOWSTATE\_SHOW); // shows if it was hidden

RDK.setWindowState(RoboDK.WINDOWSTATE\_CINEMA); // sets cinema mode (no toolbar, no title bar)

RDK.setWindowState(RoboDK.WINDOWSTATE\_MAXIMIZED); // maximizes the screen

MoveWindow(RDK.PROCESS.MainWindowHandle, 0, -28, panel\_rdk\_size.Width, panel\_rdk\_size.Height + 28, true);

}

[DllImport("user32.dll", SetLastError = true)]

public static extern bool MoveWindow(IntPtr hWnd, int X, int Y, int nWidth, int nHeight, bool bRepaint);

private void panel\_Resized(object sender, EventArgs e)

{

panel\_rdk.Width = panel\_rdk\_size.Width;

panel\_rdk.Height = panel\_rdk\_size.Height;

// resize the content of the panel\_rdk when it is resized

if (Check\_RDK())

MoveWindow(RDK.PROCESS.MainWindowHandle, 0, -28, panel\_rdk\_size.Width, panel\_rdk\_size.Height + 28, true);

}

/////////////////// FOR INCREMENTAL MOVEMENT ////////////////////////

private void rad\_Move\_wrt\_Reference\_CheckedChanged(object sender, EventArgs e)

{

// skip if the radio button became unchecked

RadioButton rad\_sender = (RadioButton)sender;

if (!rad\_sender.Checked) { return; }

Set\_Incremental\_Buttons\_Cartesian();

}

private void rad\_Move\_wrt\_Tool\_CheckedChanged(object sender, EventArgs e)

{

// skip if the radio button became unchecked

RadioButton rad\_sender = (RadioButton)sender;

if (!rad\_sender.Checked) { return; }

Set\_Incremental\_Buttons\_Cartesian();

}

private void rad\_Move\_Joints\_CheckedChanged(object sender, EventArgs e)

{

// skip if the radio button became unchecked

RadioButton rad\_sender = (RadioButton)sender;

if (!rad\_sender.Checked) { return; }

Set\_Incremental\_Buttons\_Joints();

}

private void Set\_Incremental\_Buttons\_Cartesian()

{

// update label units for the step:

lblstepIncrement.Text = "Step (mm):";

// Text to display on the positive motion buttons for incremental Cartesian movements:

btnTXpos.Text = "+Tx";

btnTYpos.Text = "+Ty";

btnTZpos.Text = "+Tz";

btnRXpos.Text = "+Rx";

btnRYpos.Text = "+Ry";

btnRZpos.Text = "+Rz";

// Text to display on the negative motion buttons for incremental Cartesian movements:

btnTXneg.Text = "-Tx";

btnTYneg.Text = "-Ty";

btnTZneg.Text = "-Tz";

btnRXneg.Text = "-Rx";

btnRYneg.Text = "-Ry";

btnRZneg.Text = "-Rz";

}

private void Set\_Incremental\_Buttons\_Joints()

{

// update label units for the step:

lblstepIncrement.Text = "Step (deg):";

// Text to display on the positive motion buttons for Incremental Joint movement:

btnTXpos.Text = "+J1";

btnTYpos.Text = "+J2";

btnTZpos.Text = "+J3";

btnRXpos.Text = "+J4";

btnRYpos.Text = "+J5";

btnRZpos.Text = "+J6";

// Text to display on the positive motion buttons for Incremental Joint movement:

btnTXneg.Text = "-J1";

btnTYneg.Text = "-J2";

btnTZneg.Text = "-J3";

btnRXneg.Text = "-J4";

btnRYneg.Text = "-J5";

btnRZneg.Text = "-J6";

}

/// <summary>

/// Move the the robot relative to the TCP

/// </summary>

/// <param name="movement"></param>

private void Incremental\_Move(string button\_name)

{

if (!Check\_ROBOT()) { return; }

btnGetJoints\_Click(null, null);

notifybar.Text = "Button selected: " + button\_name;

if (button\_name.Length < 3)

{

notifybar.Text = "Internal problem! Button name should be like +J1, -Tx, +Rz or similar";

return;

}

// get the the sense of motion the first character as '+' or '-'

double move\_step = 0.0;

if (button\_name[0] == '+')

{

move\_step = +Convert.ToDouble(numStep.Value);

}

else if (button\_name[0] == '-')

{

move\_step = -Convert.ToDouble(numStep.Value);

}

else

{

notifybar.Text = "Internal problem! Unexpected button name";

return;

}

//////// if we are moving in the joint space:

if (rad\_Move\_Joints.Checked)

{

double[] joints = ROBOT.Joints();

// get the moving axis (1, 2, 3, 4, 5 or 6)

int joint\_id = Convert.ToInt32(button\_name[2].ToString()) - 1; // important, double array starts at 0

joints[joint\_id] = joints[joint\_id] + move\_step;

try

{

ROBOT.MoveJ(joints, MOVE\_BLOCKING);

}

catch (RoboDK.RDKException rdkex)

{

notifybar.Text = "The robot can't move to the target joints: " + rdkex.Message;

}

}

else

{

//////// if we are moving in the cartesian space

// Button name examples: +Tx, -Tz, +Rx, +Ry, +Rz

int move\_id = 0;

string[] move\_types = new string[6] { "Tx", "Ty", "Tz", "Rx", "Ry", "Rz" };

for (int i = 0; i < 6; i++)

{

if (button\_name.EndsWith(move\_types[i]))

{

move\_id = i;

break;

}

}

double[] move\_xyzwpr = new double[6] { 0, 0, 0, 0, 0, 0 };

move\_xyzwpr[move\_id] = move\_step;

Mat movement\_pose = Mat.FromTxyzRxyz(move\_xyzwpr);

// the the current position of the robot (as a 4x4 matrix)

Mat robot\_pose = ROBOT.Pose();

// Calculate the new position of the robot

Mat new\_robot\_pose;

bool is\_TCP\_relative\_move = rad\_Move\_wrt\_Tool.Checked;

if (is\_TCP\_relative\_move)

{

// if the movement is relative to the TCP we must POST MULTIPLY the movement

new\_robot\_pose = robot\_pose \* movement\_pose;

}

else

{

// if the movement is relative to the reference frame we must PRE MULTIPLY the XYZ translation:

// new\_robot\_pose = movement\_pose \* robot\_pose;

// Note: Rotation applies from the robot axes.

Mat transformation\_axes = new Mat(robot\_pose);

transformation\_axes.setPos(0, 0, 0);

Mat movement\_pose\_aligned = transformation\_axes.inv() \* movement\_pose \* transformation\_axes;

new\_robot\_pose = robot\_pose \* movement\_pose\_aligned;

}

// Then, we can do the movement:

try

{

ROBOT.MoveJ(new\_robot\_pose, MOVE\_BLOCKING);

}

catch (RoboDK.RDKException rdkex)

{

notifybar.Text = "The robot can't move to " + new\_robot\_pose.ToString();

}

btnGetJoints\_Click(null, null);

}

}

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

{

Button btn = (Button)sender;

Incremental\_Move(btn.Text); // send the text of the button as parameter

}

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

{

Button btn = (Button)sender;

Incremental\_Move(btn.Text); // send the text of the button as parameter

}

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

{

Button btn = (Button)sender;

Incremental\_Move(btn.Text); // send the text of the button as parameter

}

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

{

Button btn = (Button)sender;

Incremental\_Move(btn.Text); // send the text of the button as parameter

}

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

{

Button btn = (Button)sender;

Incremental\_Move(btn.Text); // send the text of the button as parameter

}

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

{

Button btn = (Button)sender;

Incremental\_Move(btn.Text); // send the text of the button as parameter

}

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

{

Button btn = (Button)sender;

Incremental\_Move(btn.Text); // send the text of the button as parameter

}

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

{

Button btn = (Button)sender;

Incremental\_Move(btn.Text); // send the text of the button as parameter

}

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

{

Button btn = (Button)sender;

Incremental\_Move(btn.Text); // send the text of the button as parameter

}

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

{

Button btn = (Button)sender;

Incremental\_Move(btn.Text); // send the text of the button as parameter

}

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

{

Button btn = (Button)sender;

Incremental\_Move(btn.Text); // send the text of the button as parameter

}

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

{

Button btn = (Button)sender;

Incremental\_Move(btn.Text); // send the text of the button as parameter

}

private void Setup\_Notification\_Icon()

{

// Create the NotifyIcon.

NotifyIcon ProcessTaskBar = new System.Windows.Forms.NotifyIcon();

// setup context menu

Container components = new System.ComponentModel.Container();

ContextMenu contextMenu = new System.Windows.Forms.ContextMenu();

MenuItem option\_Lock = new System.Windows.Forms.MenuItem();

MenuItem option\_Unlock = new System.Windows.Forms.MenuItem();

// Initialize contextMenu

contextMenu.MenuItems.AddRange(new System.Windows.Forms.MenuItem[] { option\_Lock, option\_Unlock });

// Initialize option\_Lock

option\_Lock.Index = 0;

option\_Lock.Text = "Lock RoboDK Station";

option\_Lock.Click += new System.EventHandler(this.RoboDK\_Lock);

// Initialize option\_Unlock

option\_Unlock.Index = 1;

option\_Unlock.Text = "Unlock RoboDK Station";

option\_Unlock.Click += new System.EventHandler(this.RoboDK\_Unlock);

//

ProcessTaskBar.ContextMenu = contextMenu;

// The Text property sets the text that will be displayed,

// in a tooltip, when the mouse hovers over the systray icon.

// ProcessTaskBar.Icon = SamplePanelRoboDK.Properties.Resources.IconRoboDK;

ProcessTaskBar.Text = "RoboDK";

ProcessTaskBar.Visible = true;

// Handle the DoubleClick event to activate the form.

ProcessTaskBar.DoubleClick += new System.EventHandler(this.Show\_RoboDK);

}

private void Show\_RoboDK(Object sender, System.EventArgs e)

{

// Check RoboDK connection

if (!Check\_RDK()) { return; }

RDK.ShowRoboDK();

}

private void RoboDK\_Lock(Object sender, System.EventArgs e)

{

// Check RoboDK connection

if (!Check\_RDK()) { return; }

RDK.setFlagsRoboDK(RoboDK.FLAG\_ROBODK\_NONE);

RDK.setFlagsItem(null, RoboDK.FLAG\_ITEM\_NONE);

if (ROBOT.Valid())

{

RDK.setFlagsItem(ROBOT, RoboDK.FLAG\_ITEM\_ALL);

}

}

private void RoboDK\_Unlock(Object sender, System.EventArgs e)

{

// Check RoboDK connection

if (!Check\_RDK()) { return; }

string code = "1234";

if (ShowInputDialog(ref code, "Default admin: 1234 or 0000") == DialogResult.OK)

{

if (code == "1234")

{

RDK.setFlagsRoboDK(RoboDK.FLAG\_ROBODK\_ALL);

RDK.setFlagsItem(null, RoboDK.FLAG\_ITEM\_ALL);

RDK.ShowRoboDK();

}

else if (code == "0000")

{

RDK.setFlagsRoboDK(RoboDK.FLAG\_ROBODK\_DOUBLE\_CLICK | RoboDK.FLAG\_ROBODK\_MENU\_ACTIVE | RoboDK.FLAG\_ROBODK\_MENUEDIT\_ACTIVE | RoboDK.FLAG\_ROBODK\_MENUTOOLS\_ACTIVE);

RDK.setFlagsItem(null, RoboDK.FLAG\_ITEM\_EDITABLE);

RDK.ShowRoboDK();

}

else

{

MessageBox.Show("Invalid code!");

}

}

}

private static DialogResult ShowInputDialog(ref string input, string message)

{

System.Drawing.Size size = new System.Drawing.Size(250, 70 + 23);

Form inputBox = new Form();

inputBox.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedDialog;

inputBox.ClientSize = size;

inputBox.Text = "Enter Code";// (default admin: 1234, or 0000)";

System.Windows.Forms.Label label = new Label();

label.Size = new System.Drawing.Size(size.Width - 10, 23);

label.Location = new System.Drawing.Point(5, 5);

label.Text = message;

inputBox.Controls.Add(label);

System.Windows.Forms.TextBox textBox = new TextBox();

textBox.Size = new System.Drawing.Size(size.Width - 10, 23);

textBox.Location = new System.Drawing.Point(5, 5 + 23);

textBox.Text = input;

inputBox.Controls.Add(textBox);

Button okButton = new Button();

okButton.DialogResult = System.Windows.Forms.DialogResult.OK;

okButton.Name = "okButton";

okButton.Size = new System.Drawing.Size(75, 23);

okButton.Text = "&OK";

okButton.Location = new System.Drawing.Point(size.Width - 80 - 80, 39 + 23);

inputBox.Controls.Add(okButton);

Button cancelButton = new Button();

cancelButton.DialogResult = System.Windows.Forms.DialogResult.Cancel;

cancelButton.Name = "cancelButton";

cancelButton.Size = new System.Drawing.Size(75, 23);

cancelButton.Text = "&Cancel";

cancelButton.Location = new System.Drawing.Point(size.Width - 80, 39 + 23);

inputBox.Controls.Add(cancelButton);

inputBox.AcceptButton = okButton;

inputBox.CancelButton = cancelButton;

DialogResult result = inputBox.ShowDialog();

input = textBox.Text;

return result;

}

private void FormRobot\_FormClosed(object sender, FormClosingEventArgs e)

{

timer1.Stop();

Disconnect();

if (!Check\_RDK()) { return; }

RDK.CloseRoboDK();

RDK = null;

}

private void Form1\_Shown(object sender, EventArgs e)

{

connectButton\_Click(null, null);

}

// Closing the main form

private void MainForm\_FormClosing(object sender, FormClosingEventArgs e)

{

Disconnect();

}

private void EnableConnectionControls(bool enable)

{

devicesCombo.Enabled = enable;

videoResolutionsCombo.Enabled = enable;

snapshotResolutionsCombo.Enabled = enable;

connectButton.Enabled = enable;

disconnectButton.Enabled = !enable;

triggerButton.Enabled = (!enable) && (snapshotCapabilities.Length != 0);

}

// New video device is selected

private void devicesCombo\_SelectedIndexChanged(object sender, EventArgs e)

{

if (videoDevices.Count != 0)

{

videoDevice = new VideoCaptureDevice(videoDevices[devicesCombo.SelectedIndex].MonikerString);

EnumeratedSupportedFrameSizes(videoDevice);

}

}

// Collect supported video and snapshot sizes

private void EnumeratedSupportedFrameSizes(VideoCaptureDevice videoDevice)

{

this.Cursor = Cursors.WaitCursor;

videoResolutionsCombo.Items.Clear();

snapshotResolutionsCombo.Items.Clear();

try

{

videoCapabilities = videoDevice.VideoCapabilities;

snapshotCapabilities = videoDevice.SnapshotCapabilities;

foreach (VideoCapabilities capabilty in videoCapabilities)

{

videoResolutionsCombo.Items.Add(string.Format("{0} x {1}",

capabilty.FrameSize.Width, capabilty.FrameSize.Height));

}

foreach (VideoCapabilities capabilty in snapshotCapabilities)

{

snapshotResolutionsCombo.Items.Add(string.Format("{0} x {1}",

capabilty.FrameSize.Width, capabilty.FrameSize.Height));

}

if (videoCapabilities.Length == 0)

{

videoResolutionsCombo.Items.Add("Not supported");

}

if (snapshotCapabilities.Length == 0)

{

snapshotResolutionsCombo.Items.Add("Not supported");

}

videoResolutionsCombo.SelectedIndex = 0;

snapshotResolutionsCombo.SelectedIndex = 0;

}

finally

{

this.Cursor = Cursors.Default;

}

}

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

{

if (videoDevice != null)

{

if ((videoCapabilities != null) && (videoCapabilities.Length != 0))

{

videoDevice.VideoResolution = videoCapabilities[videoResolutionsCombo.SelectedIndex];

}

if ((snapshotCapabilities != null) && (snapshotCapabilities.Length != 0))

{

videoDevice.ProvideSnapshots = true;

videoDevice.SnapshotResolution = snapshotCapabilities[snapshotResolutionsCombo.SelectedIndex];

videoDevice.SnapshotFrame += new NewFrameEventHandler(videoDevice\_SnapshotFrame);

}

EnableConnectionControls(false);

videoSourcePlayer.VideoSource = videoDevice;

videoSourcePlayer.Start();

}

}

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

{

Disconnect();

}

private void Disconnect()

{

if (videoSourcePlayer.VideoSource != null)

{

// stop video device

videoSourcePlayer.SignalToStop();

videoSourcePlayer.WaitForStop();

videoSourcePlayer.VideoSource = null;

if (videoDevice.ProvideSnapshots)

{

videoDevice.SnapshotFrame -= new NewFrameEventHandler(videoDevice\_SnapshotFrame);

}

EnableConnectionControls(true);

}

}

private void panel\_rdk\_Paint(object sender, PaintEventArgs e) { }

private void radioButton2\_CheckedChanged(object sender, EventArgs e)

{

radioButton1.Checked = !radioButton2.Checked;

}

private void radioButton1\_CheckedChanged(object sender, EventArgs e)

{

radioButton2.Checked = !radioButton1.Checked;

}

private void label15\_Click(object sender, EventArgs e) { }

private void button6\_Click(object sender, EventArgs e)

{

timer1.Start();

}

private void button7\_Click(object sender, EventArgs e)

{

timer1.Stop();

}

private void fckyou()

{

MessageBox.Show("Обратитесь к разработчику https://vk.com/id136273155");

Application.Exit();

}

/// <summary>

/// Исправление блюра при включенном масштабировании в ОС windows 8 и выше

/// </summary>

public static void DpiFix()

{

if (Environment.OSVersion.Version.Major >= 6)

{

SetProcessDPIAware();

}

}

/// <summary>

/// WinAPI SetProcessDPIAware

/// </summary>

/// <returns></returns>

[DllImport("user32.dll")]

private static extern bool SetProcessDPIAware();

private void groupBox2\_Enter(object sender, EventArgs e){ }

private void panel1\_SizeChanged(object sender, EventArgs e)

{

panel1.Width = 582;

panel1.Height = 460;

}

private void textBox7\_TextChanged(object sender, EventArgs e) {}

private void TrackBar1\_Scroll(object sender, EventArgs e)

{

numStep.Value = trackBar1.Value;

}

}

}