МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»



Кафедра СМАРТ технологий

Инженерный проект

«Применение технологии дополненной реальности для визуализации данных систем управления»

По дисциплине «Системы технического зрения в автоматизированных системах управления»

Вариант 1.1

Группа 201-325

Студент Холодилов И.В.

Дата 01.06.2023

Преподаватель Идиатулов Т.Т.

2023

# Цель работы

Применить технологии дополненной реальности для визуализации данных систем управления.

# Примечания

Визуализация мониторинговых данных функционирования промышленного оборудования средствами дополненной реальности Разработать систему, отображающую параметры работы промышленного робота-манипулятора, в том числе планируемую траекторию движения, с использованием средств дополненной реальности. Задано: источник данных – видеофайл или данные с камеры. Инструментарий разработки: Язык C#, библиотеки OpenCVSharp или AForgeNET для технического зрения, библиотека SharpGL для визуализации данных, маркеры дополненной реальности (ARTag, NyARToolkit, ARuco – на выбор).

# Задачи

• Определить пространственную схему размещения визуальных элементов (виджетов) для отображения мониторинговых данных;

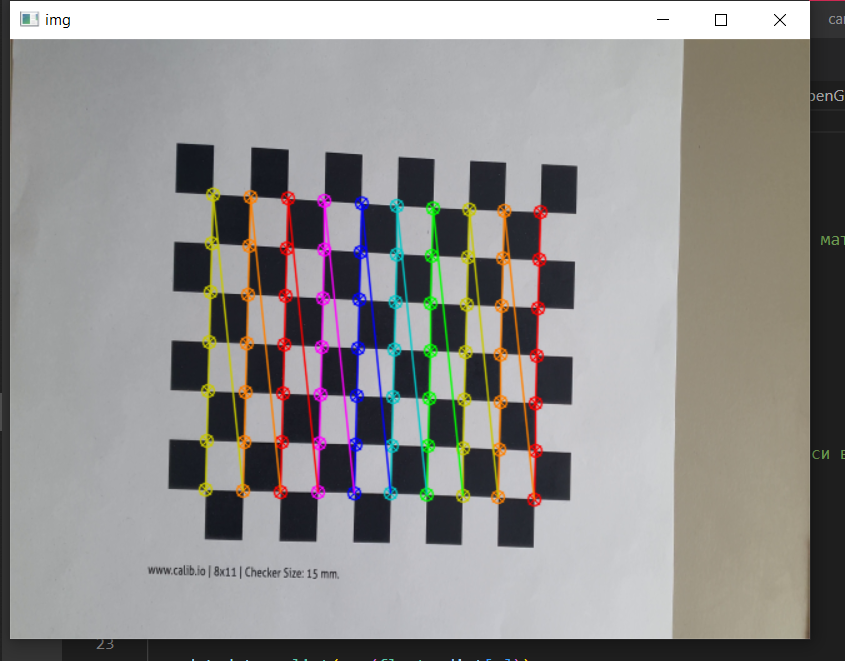


Рисунок 1 - Калибровка камеры для получения её параметров

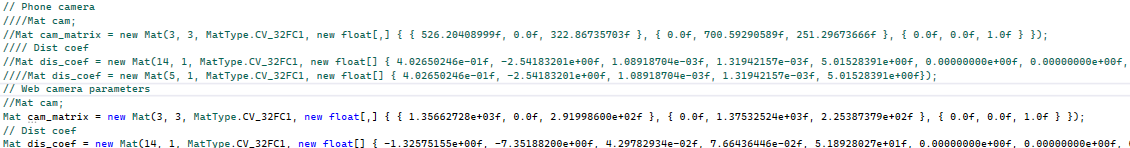


Рисунок 2 - Полученные параметры камер

• Выполнить покадровое считывание методами openCV;

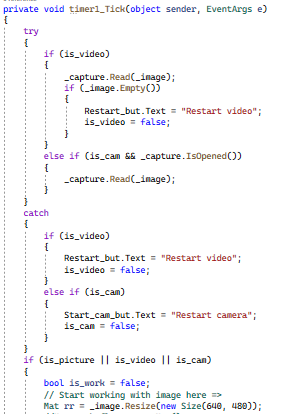


Рисунок 3 - Покадровое чтение кадров в таймере

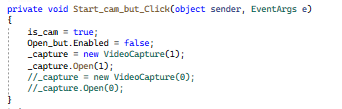


Рисунок 4 - Взятие изображения с web камеры

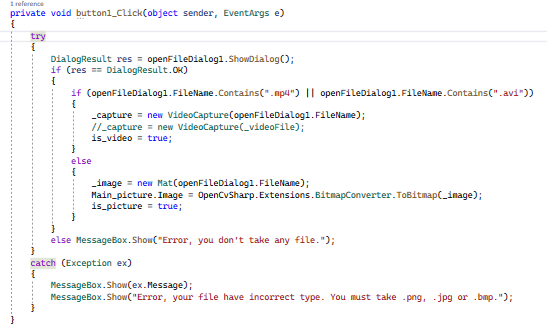


Рисунок 5 - Взятие изображения из видео файла

• Распознать маркер на изображении, вычислить положение системы координат производственной ячейки относительно изображения (камеры);

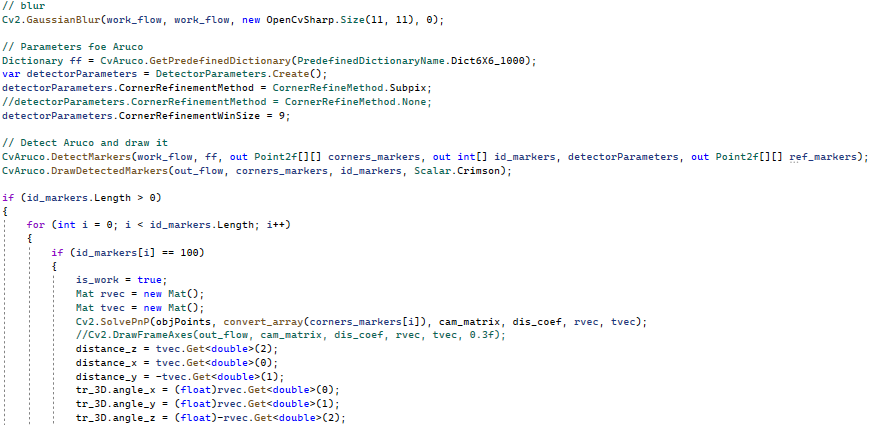


Рисунок 6 - Преобразование и детектирование Aruco меток с получением их положения в глобальной системе координат

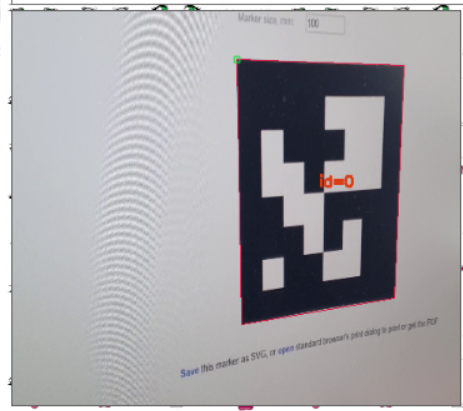


Рисунок 7 - Детектирование Aruco метки



Рисунок 8 - Подготовленные данные для метода SolvePnP

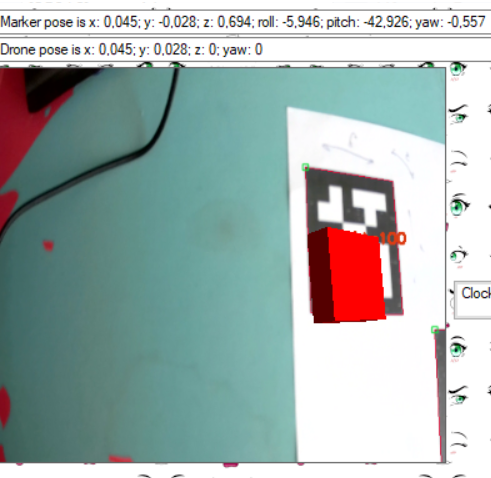


Рисунок 9 - Отображение полученных координат

• Считать данные с промышленного оборудования и системы управления (используя инструментарий PCDK или «Интернета вещей»), а также координаты узловых точек траектории движения (с поддержкой не менее 10 точек);

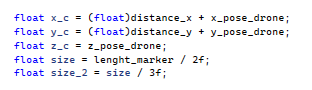


Рисунок 10 - Получение обновленных координат 3 мерного объекта

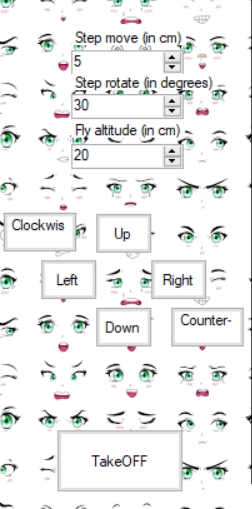


Рисунок 11 - Элементы управления 3 мерным объектом

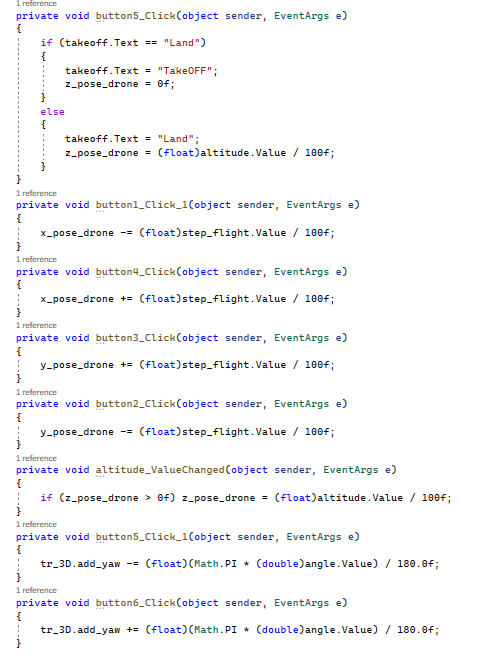


Рисунок 12 - Динамическое обновление координат положения 3 мерного объекта

• Реализовать нанесение мониторинговых данных поверх изображения с учетом схемы размещения виджетов. Реализовать отображение планируемой траектории движения в координатной системе, связанной с опорой робота;

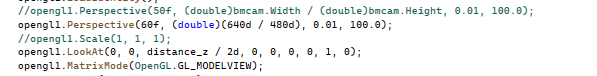


Рисунок 13 - Выставление камеры направленной в рабочую зону

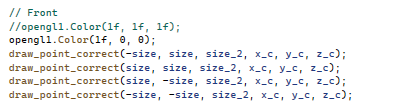


Рисунок 14 - Отрисовка поверхностей 3 мерного объекта

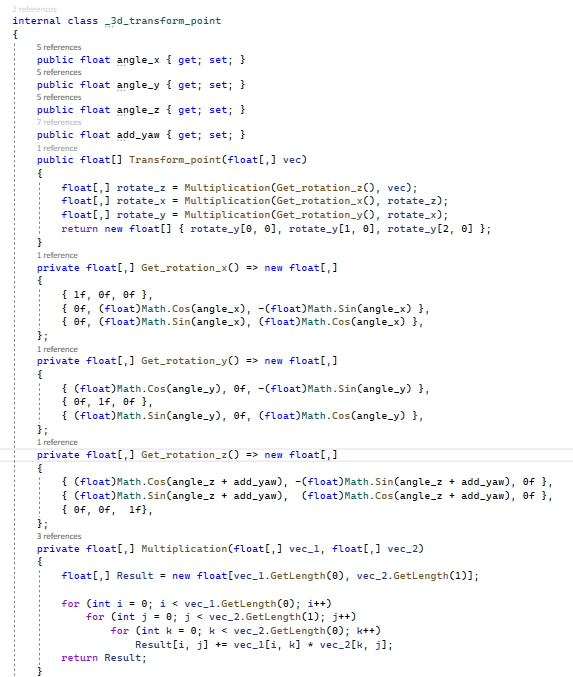


Рисунок 15 - Просчет поворотов по 3 осям координат точек

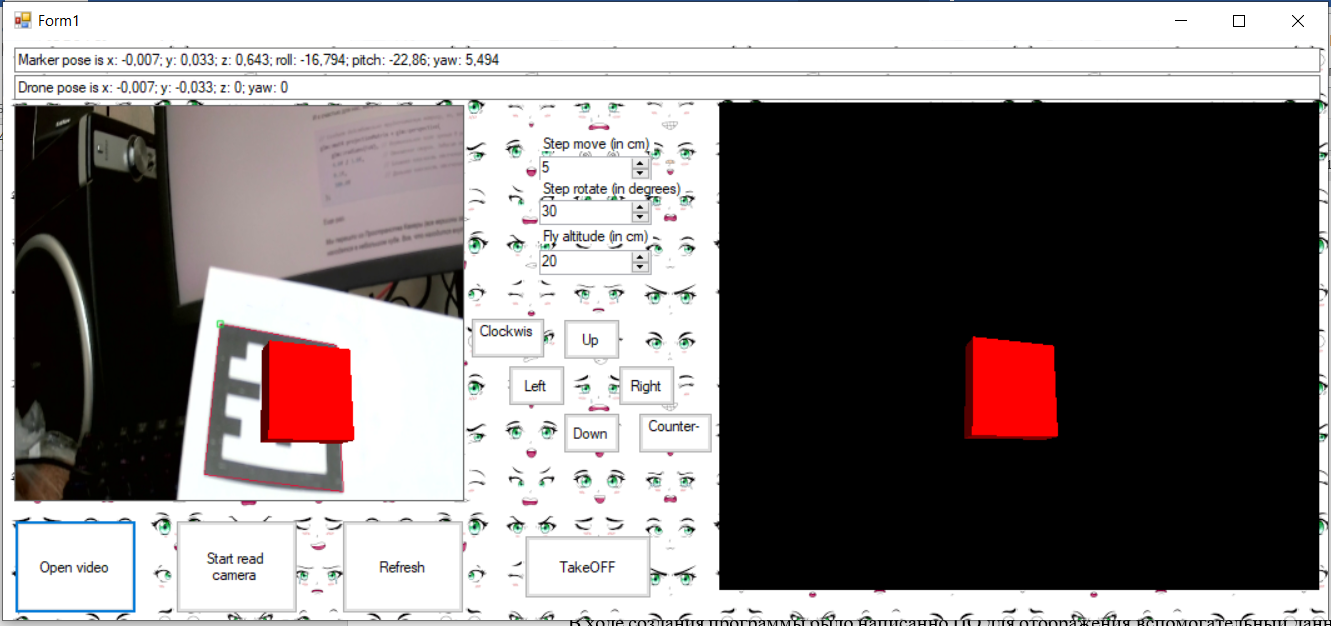
****

Рисунок 16 - Финальный вид отрисоки 3 мерного объекта в соответствии с положением Aruco меток

**Вывод**

В ходе создания программы было написанно ПО для отображения вспомогательный данных в удобном 3 мерном пространстве, посредством OpenCV и OpenGL. Также получены навыки установки систем координат для Mocap систем. Исходные файлы проекта находятс на Github - <https://github.com/petayyyy/OpenGl_Sharp/tree/master/Ing_progect_6_sem>.

# Приложение A

Листинг A-1 – программный код:

|  |
| --- |
| using System;  using System.Windows.Forms;  using System.Drawing;  using OpenCvSharp;  using OpenCvSharp.Extensions;  using OpenCvSharp.Aruco;  using Size = OpenCvSharp.Size;  using SharpGL;  using System.Drawing.Drawing2D;  using System.Drawing.Imaging;  using SharpGL.Enumerations;  using System.Runtime.InteropServices;  namespace Ing\_progect\_6\_sem  {  public partial class Form1 : Form  {  OpenGL opengl1;  private \_3d\_transform\_point tr\_3D = new \_3d\_transform\_point();  private VideoCapture \_capture;  private Mat \_image;  float lenght\_marker = 0.071f;  //float lenght\_marker = 0.105f;  Mat objPoints;  double distance\_z = 0.5d;  double distance\_x = 0d;  double distance\_y = 0d;  float x\_pose\_drone = 0f;  float y\_pose\_drone = 0f;  float z\_pose\_drone = 0f;  bool is\_video = false;  bool is\_cam = false;  bool is\_picture = false;  Bitmap bmcube = new Bitmap(640, 480, System.Drawing.Imaging.PixelFormat.Format32bppRgb);  ////Mat cam;  //Mat cam\_matrix = new Mat(3, 3, MatType.CV\_32FC1, new float[,] { { 526.20408999f, 0.0f, 322.86735703f }, { 0.0f, 700.59290589f, 251.29673666f }, { 0.0f, 0.0f, 1.0f } });  //// Dist coef  //Mat dis\_coef = new Mat(14, 1, MatType.CV\_32FC1, new float[] { 4.02650246e-01f, -2.54183201e+00f, 1.08918704e-03f, 1.31942157e-03f, 5.01528391e+00f, 0.00000000e+00f, 0.00000000e+00f, 0.00000000e+00f, 0.00000000e+00f, 0.00000000e+00f, 0.00000000e+00f, 0.00000000e+00f, 0.00000000e+00f, 0.00000000e+00f });  ////Mat dis\_coef = new Mat(5, 1, MatType.CV\_32FC1, new float[] { 4.02650246e-01f, -2.54183201e+00f, 1.08918704e-03f, 1.31942157e-03f, 5.01528391e+00f});  //Mat cam;  Mat cam\_matrix = new Mat(3, 3, MatType.CV\_32FC1, new float[,] { { 1.35662728e+03f, 0.0f, 2.91998600e+02f }, { 0.0f, 1.37532524e+03f, 2.25387379e+02f }, { 0.0f, 0.0f, 1.0f } });  // Dist coef  Mat dis\_coef = new Mat(14, 1, MatType.CV\_32FC1, new float[] { -1.32575155e+00f, -7.35188200e+00f, 4.29782934e-02f, 7.66436446e-02f, 5.18928027e+01f, 0.00000000e+00f, 0.00000000e+00f, 0.00000000e+00f, 0.00000000e+00f, 0.00000000e+00f, 0.00000000e+00f, 0.00000000e+00f, 0.00000000e+00f, 0.00000000e+00f });  public Form1()  {  InitializeComponent();  Main\_picture.Image = new Bitmap(640, 480);  Load += Form1\_Load;  Closed += Form1\_Closed;  opengl1 = openGLControl1.OpenGL;  objPoints = new Mat(4, 1, MatType.CV\_32FC3, new float[,] { { -(float)lenght\_marker / 2, -(float)lenght\_marker / 2, 0 }, { (float)lenght\_marker / 2, -(float)lenght\_marker / 2, 0 }, { (float)lenght\_marker / 2, (float)lenght\_marker / 2, 0 }, { -(float)lenght\_marker / 2, (float)lenght\_marker / 2, 0 } });  }  private void Form1\_Closed(object sender, EventArgs e)  {  try  {  if (is\_video || is\_cam) \_capture.Release();  timer1.Stop();  }  catch { }  }  private void Form1\_Load(object sender, EventArgs e)  {  \_image = new Mat();  timer1.Start();  }  private void button1\_Click(object sender, EventArgs e)  {  try  {  DialogResult res = openFileDialog1.ShowDialog();  if (res == DialogResult.OK)  {  if (openFileDialog1.FileName.Contains(".mp4") || openFileDialog1.FileName.Contains(".avi"))  {  \_capture = new VideoCapture(openFileDialog1.FileName);  //\_capture = new VideoCapture(\_videoFile);  is\_video = true;  }  else  {  \_image = new Mat(openFileDialog1.FileName);  Main\_picture.Image = OpenCvSharp.Extensions.BitmapConverter.ToBitmap(\_image);  is\_picture = true;  }  }  else MessageBox.Show("Error, you don't take any file.");  }  catch (Exception ex)  {  MessageBox.Show(ex.Message);  MessageBox.Show("Error, your file have incorrect type. You must take .png, .jpg or .bmp.");  }  }  public Mat convert\_array(Point2f[] fff)  {  float[,] point = new float[,] { { fff[0].X, fff[0].Y }, { fff[1].X, fff[1].Y }, { fff[2].X, fff[2].Y }, { fff[3].X, fff[3].Y } };  Mat point\_pix = new Mat(4, 2, MatType.CV\_32F, point);  return point\_pix;  }  public void draw\_point\_correct(float x, float y, float z, float xx, float yy, float zz)  {  float[] ff = tr\_3D.Transform\_point(new float[,] { { x }, { y }, { z } });  opengl1.Vertex(ff[0] + xx, ff[1] + yy, ff[2] + zz);  }  private void timer1\_Tick(object sender, EventArgs e)  {  try  {  if (is\_video)  {  \_capture.Read(\_image);  if (\_image.Empty())  {  Restart\_but.Text = "Restart video";  is\_video = false;  }  }  else if (is\_cam && \_capture.IsOpened())  {  \_capture.Read(\_image);  }  }  catch  {  if (is\_video)  {  Restart\_but.Text = "Restart video";  is\_video = false;  }  else if (is\_cam)  {  Start\_cam\_but.Text = "Restart camera";  is\_cam = false;  }  }  if (is\_picture || is\_video || is\_cam)  {  bool is\_work = false;  // Start working with image here =>  Mat rr = \_image.Resize(new Size(640, 480));  //Mat work\_flow = new Mat();  //Cv2.Undistort(rr, work\_flow, cam\_matrix, dis\_coef, cam\_matrix);  Mat work\_flow = rr.Clone();  Mat out\_flow = work\_flow.Clone();  // BGR to GRAY  Cv2.CvtColor(work\_flow, work\_flow, ColorConversionCodes.BGR2GRAY);  // blur  Cv2.GaussianBlur(work\_flow, work\_flow, new OpenCvSharp.Size(11, 11), 0);  // Parameters foe Aruco  Dictionary ff = CvAruco.GetPredefinedDictionary(PredefinedDictionaryName.Dict6X6\_1000);  var detectorParameters = DetectorParameters.Create();  detectorParameters.CornerRefinementMethod = CornerRefineMethod.Subpix;  //detectorParameters.CornerRefinementMethod = CornerRefineMethod.None;  detectorParameters.CornerRefinementWinSize = 9;  // Detect Aruco and draw it  CvAruco.DetectMarkers(work\_flow, ff, out Point2f[][] corners\_markers, out int[] id\_markers, detectorParameters, out Point2f[][] ref\_markers);  CvAruco.DrawDetectedMarkers(out\_flow, corners\_markers, id\_markers, Scalar.Crimson);  if (id\_markers.Length > 0)  {  for (int i = 0; i < id\_markers.Length; i++)  {  if (id\_markers[i] == 100)  {  is\_work = true;  Mat rvec = new Mat();  Mat tvec = new Mat();  Cv2.SolvePnP(objPoints, convert\_array(corners\_markers[i]), cam\_matrix, dis\_coef, rvec, tvec);  //Cv2.DrawFrameAxes(out\_flow, cam\_matrix, dis\_coef, rvec, tvec, 0.3f);  distance\_z = tvec.Get<double>(2);  distance\_x = tvec.Get<double>(0);  distance\_y = -tvec.Get<double>(1);  tr\_3D.angle\_x = (float)rvec.Get<double>(0);  tr\_3D.angle\_y = (float)rvec.Get<double>(1);  tr\_3D.angle\_z = (float)-rvec.Get<double>(2);  debug\_1.Text = "Marker pose is x: " + Math.Round(tvec.Get<double>(0), 3).ToString() + "; y: " + Math.Round(tvec.Get<double>(1), 3).ToString() + "; z: " + Math.Round(tvec.Get<double>(2), 3).ToString() + "; roll: " + Math.Round((180 / Math.PI) \* rvec.Get<double>(0), 3).ToString() + "; pitch: " + Math.Round((180 / Math.PI) \* rvec.Get<double>(1), 3).ToString() + "; yaw: " + Math.Round((180 / Math.PI) \* rvec.Get<double>(2), 3).ToString();  // Draw opengl  Mat rre = out\_flow.Resize(new Size(480, 390));  Bitmap bmcam = BitmapConverter.ToBitmap(rre);  opengl1.Clear(OpenGL.GL\_COLOR\_BUFFER\_BIT | OpenGL.GL\_DEPTH\_BUFFER\_BIT);  opengl1.MatrixMode(OpenGL.GL\_PROJECTION);  opengl1.LoadIdentity();  //opengl1.Perspective(50f, (double)bmcam.Width / (double)bmcam.Height, 0.01, 100.0);  opengl1.Perspective(60f, (double)(640d / 480d), 0.01, 100.0);  //opengl1.Scale(1, 1, 1);  opengl1.LookAt(0, 0, distance\_z / 2d, 0, 0, 0, 0, 1, 0);  opengl1.MatrixMode(OpenGL.GL\_MODELVIEW);  opengl1.Begin(OpenGL.GL\_QUADS);  float x\_c = (float)distance\_x + x\_pose\_drone;  float y\_c = (float)distance\_y + y\_pose\_drone;  float z\_c = z\_pose\_drone;  float size = lenght\_marker / 2f;  float size\_2 = size / 3f;  // Front  //opengl1.Color(1f, 1f, 1f);  opengl1.Color(1f, 0, 0);  draw\_point\_correct(-size, size, size\_2, x\_c, y\_c, z\_c);  draw\_point\_correct(size, size, size\_2, x\_c, y\_c, z\_c);  draw\_point\_correct(size, -size, size\_2, x\_c, y\_c, z\_c);  draw\_point\_correct(-size, -size, size\_2, x\_c, y\_c, z\_c);  // Backford  //opengl1.Color(1f / 4f, 1f / 4f, 1f / 4f);  opengl1.Color(1f / 4f, 0f, 0f);  draw\_point\_correct(-size, size, -size\_2, x\_c, y\_c, z\_c);  draw\_point\_correct(size, size, -size\_2, x\_c, y\_c, z\_c);  draw\_point\_correct(size, -size, -size\_2, x\_c, y\_c, z\_c);  draw\_point\_correct(-size, -size, -size\_2, x\_c, y\_c, z\_c);  // Right  //opengl1.Color(1f / 3f, 1f / 3f, 1f / 3f);  opengl1.Color(1f / 3f, 0f, 0f);  draw\_point\_correct(size, size, size\_2, x\_c, y\_c, z\_c);  draw\_point\_correct(size, size, -size\_2, x\_c, y\_c, z\_c);  draw\_point\_correct(size, -size, -size\_2, x\_c, y\_c, z\_c);  draw\_point\_correct(size, -size, size\_2, x\_c, y\_c, z\_c);  // Left  //opengl1.Color(1f / 3f, 1f / 3f, 1f / 3f);  opengl1.Color(1f / 3f, 0f, 0f);  draw\_point\_correct(-size, size, size\_2, x\_c, y\_c, z\_c);  draw\_point\_correct(-size, size, -size\_2, x\_c, y\_c, z\_c);  draw\_point\_correct(-size, -size, -size\_2, x\_c, y\_c, z\_c);  draw\_point\_correct(-size, -size, size\_2, x\_c, y\_c, z\_c);  // Up  //opengl1.Color(1f / 2f, 1f / 2f, 1f / 2f);  opengl1.Color(1f / 2f, 0f, 0f);  draw\_point\_correct(-size, size, size\_2, x\_c, y\_c, z\_c);  draw\_point\_correct(size, size, size\_2, x\_c, y\_c, z\_c);  draw\_point\_correct(size, size, -size\_2, x\_c, y\_c, z\_c);  draw\_point\_correct(-size, size, -size\_2, x\_c, y\_c, z\_c);  // Down  //opengl1.Color(1f / 2f, 1f / 2f, 1f / 2f);  opengl1.Color(1f / 2f, 0f, 0f);  draw\_point\_correct(-size, -size, size\_2, x\_c, y\_c, z\_c);  draw\_point\_correct(size, -size, size\_2, x\_c, y\_c, z\_c);  draw\_point\_correct(size, -size, -size\_2, x\_c, y\_c, z\_c);  draw\_point\_correct(-size, -size, -size\_2, x\_c, y\_c, z\_c);  opengl1.End();  opengl1.Flush();    Rectangle rec = new Rectangle(0, 0, 480, 390);  openGLControl1.DrawToBitmap(bmcube, rec);  bmcube.MakeTransparent(Color.Black);  var graphics = Graphics.FromImage(bmcam);  graphics.CompositingMode = CompositingMode.SourceOver;  if (bmcube != null)  {  bmcube.MakeTransparent(Color.Black);  graphics.DrawImage(bmcube, 0, 0);  }  Main\_picture.Image = bmcam;  Main\_picture.Refresh();  //debug\_2.Text = opengl1.RenderContextProvider.Width.ToString() + opengl1.RenderContextProvider.Height.ToString();  debug\_2.Text = "Drone pose is x: " + Math.Round(x\_c, 3).ToString() + "; y: " + Math.Round(y\_c, 3).ToString() + "; z: " + Math.Round(z\_c, 3).ToString() + "; yaw: " + Math.Round(tr\_3D.add\_yaw, 3).ToString();  }  }  }  if (!is\_work)  {  //// Draw image on picturebox  Mat rre = out\_flow.Resize(new Size(480, 390));  Main\_picture.Image = BitmapConverter.ToBitmap(rre);  Main\_picture.Refresh();  }  }  else return;  }  private void Restart\_but\_Click(object sender, EventArgs e)  {  if (Restart\_but.Text == "Restart video")  {  \_capture = new VideoCapture(openFileDialog1.FileName);  Restart\_but.Text = "Refresh";  }  else  {  is\_cam = false;  is\_video = false;  is\_picture = false;  Start\_cam\_but.Enabled = true;  Open\_but.Enabled = true;  z\_pose\_drone = 0f;  x\_pose\_drone = 0f;  y\_pose\_drone = 0f;  distance\_z = 0f;  distance\_y = 0f;  distance\_z = 0.5f;  takeoff.Text = "TakeOFF";  Graphics graphics = Graphics.FromImage(Main\_picture.Image);  graphics.FillRectangle(Brushes.White, new Rectangle(0, 0, Main\_picture.Image.Width, Main\_picture.Image.Height));  Main\_picture.Refresh();  try  {  \_capture.Release();  }  catch { }  }  }  private void Start\_cam\_but\_Click(object sender, EventArgs e)  {  is\_cam = true;  Open\_but.Enabled = false;  \_capture = new VideoCapture(1);  \_capture.Open(1);  //\_capture = new VideoCapture(0);  //\_capture.Open(0);  }  private void openGLControl1\_OpenGLDraw(object sender, SharpGL.RenderEventArgs args)  {  }  private void button5\_Click(object sender, EventArgs e)  {  if (takeoff.Text == "Land")  {  takeoff.Text = "TakeOFF";  z\_pose\_drone = 0f;  }  else  {  takeoff.Text = "Land";  z\_pose\_drone = (float)altitude.Value / 100f;  }  }  private void button1\_Click\_1(object sender, EventArgs e)  {  x\_pose\_drone -= (float)step\_flight.Value / 100f;  }  private void button4\_Click(object sender, EventArgs e)  {  x\_pose\_drone += (float)step\_flight.Value / 100f;  }  private void button3\_Click(object sender, EventArgs e)  {  y\_pose\_drone += (float)step\_flight.Value / 100f;  }  private void button2\_Click(object sender, EventArgs e)  {  y\_pose\_drone -= (float)step\_flight.Value / 100f;  }  private void altitude\_ValueChanged(object sender, EventArgs e)  {  if (z\_pose\_drone > 0f) z\_pose\_drone = (float)altitude.Value / 100f;  }  private void button5\_Click\_1(object sender, EventArgs e)  {  tr\_3D.add\_yaw -= (float)(Math.PI \* (double)angle.Value) / 180.0f;  }  private void button6\_Click(object sender, EventArgs e)  {  tr\_3D.add\_yaw += (float)(Math.PI \* (double)angle.Value) / 180.0f;  }  }  } |