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;

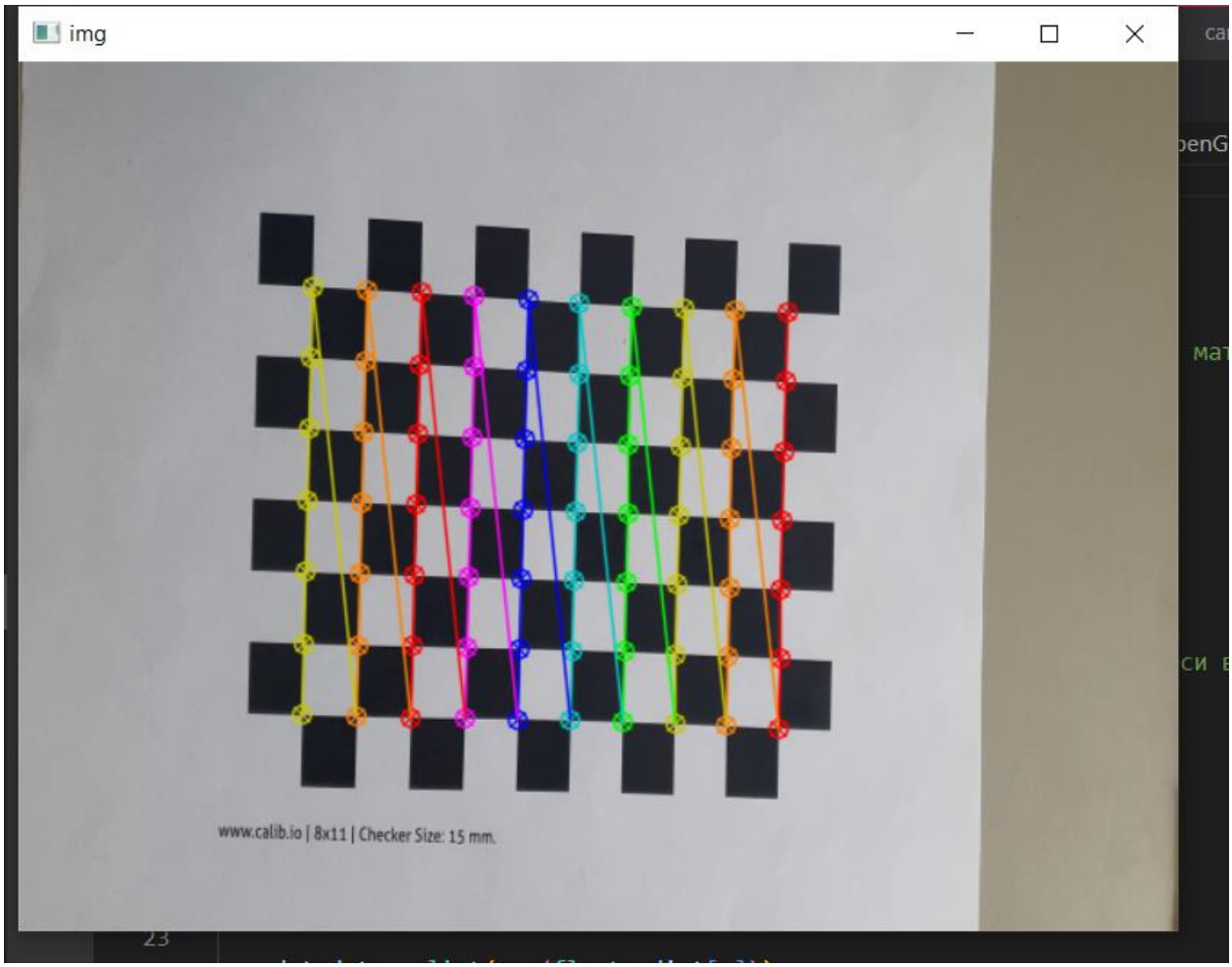

Рисунок 1 - Калибровка камеры для получения её параметров

```
// Phone camera
////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,
////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});
// Web camera parameters
//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,
```

Рисунок 2 - Полученные параметры камер

- Выполнить покадровое считывание методами openCV;

```
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));
```

Рисунок 3 - Покадровое чтение кадров в таймере

```
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);
}
```

Рисунок 4 - Взятие изображения с web камеры

```
1 reference
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.");
    }
}
```

Рисунок 5 - Взятие изображения из видео файла

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

```
// 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);
```

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

Рисунок 7 - Детектирование Aruco метки

```
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 } });
```

Рисунок 8 - Подготовленные данные для метода SolvePnP

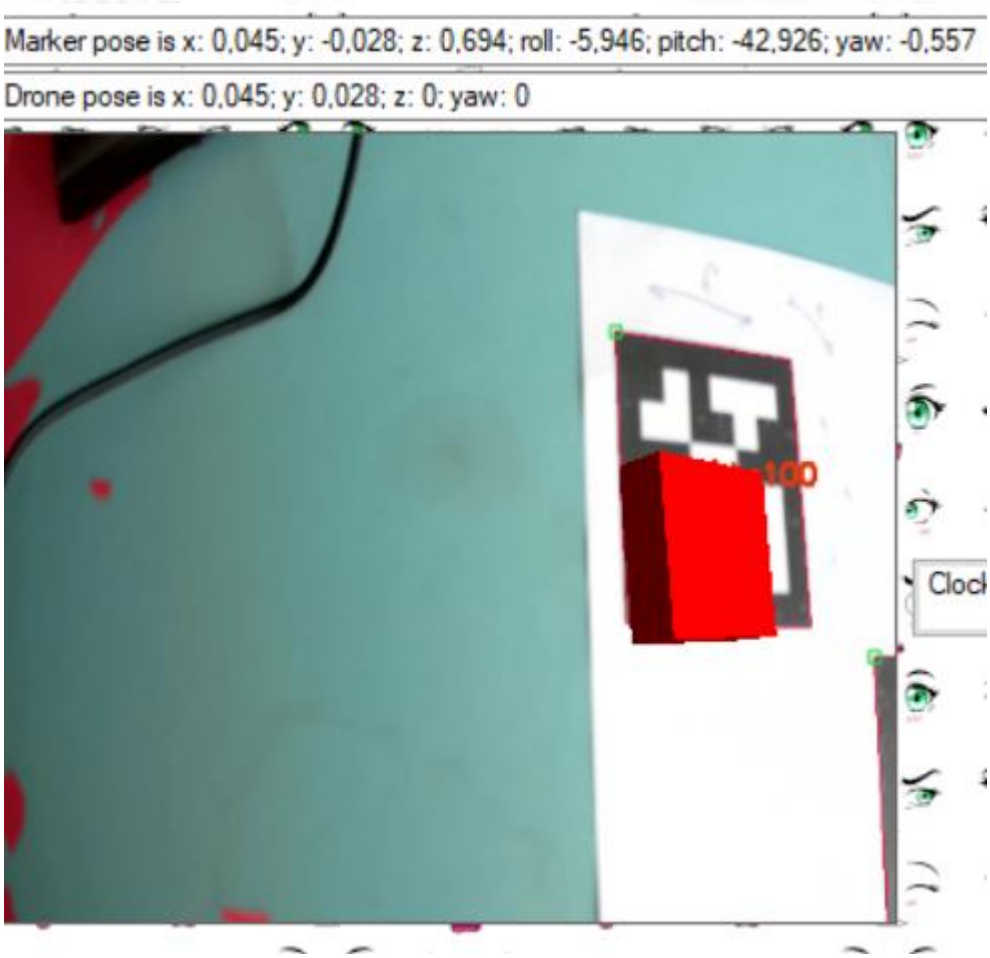

Рисунок 9 - Отображение полученных координат

• 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);

```
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;
```

Рисунок 10 - Получение обновленных координат 3 мерного объекта

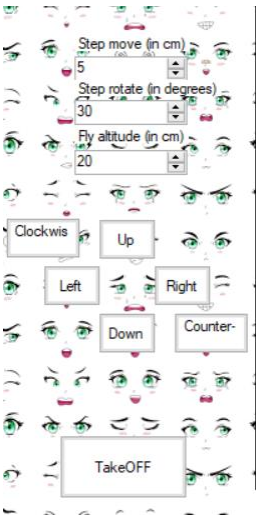

Рисунок 11 - Элементы управления 3 мерным объектом

```
1 reference
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;
    }
}
1 reference
private void button1_Click_1(object sender, EventArgs e)
{
    x_pose_drone -= (float)step_flight.Value / 100f;
}
1 reference
private void button4_Click(object sender, EventArgs e)
{
    x_pose_drone += (float)step_flight.Value / 100f;
}
1 reference
private void button3_Click(object sender, EventArgs e)
{
    y_pose_drone += (float)step_flight.Value / 100f;
}
1 reference
private void button2_Click(object sender, EventArgs e)
{
    y_pose_drone -= (float)step_flight.Value / 100f;
}
1 reference
private void altitude_ValueChanged(object sender, EventArgs e)
{
    if (z_pose_drone > 0f) z_pose_drone = (float)altitude.Value / 100f;
}
1 reference
private void button5_Click_1(object sender, EventArgs e)
{
    tr_3D.add_yaw -= (float)(Math.PI * (double)angle.Value) / 180.0f;
}
1 reference
private void button6_Click(object sender, EventArgs e)
{
    tr_3D.add_yaw += (float)(Math.PI * (double)angle.Value) / 180.0f;
}
```

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;

```
//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);
```

Рисунок 13 - Выставление камеры направленной в рабочую зону

```
// 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);
```

Рисунок 14 - Отрисовка поверхностей 3 мерного объекта

```
2 references
internal class _3d_transform_point
{
    5 references
    public float angle_x { get; set; }
    5 references
    public float angle_y { get; set; }
    5 references
    public float angle_z { get; set; }
    7 references
    public float add_yaw { get; set; }
    1 reference
    public float[] Transform_point(float[,] vec)
    {
        float[,] rotate_z = Multiplication(Get_rotation_z(), vec);
        float[,] rotate_x = Multiplication(Get_rotation_x(), rotate_z);
        float[,] rotate_y = Multiplication(Get_rotation_y(), rotate_x);
        return new float[] { rotate_y[0, 0], rotate_y[1, 0], rotate_y[2, 0] };
    }
    1 reference
    private float[,] Get_rotation_x() => new float[,]
    {
        { 1f, 0f, 0f },
        { 0f, (float)Math.Cos(angle_x), -(float)Math.Sin(angle_x) },
        { 0f, (float)Math.Sin(angle_x), (float)Math.Cos(angle_x) },
    };
    1 reference
    private float[,] Get_rotation_y() => new float[,]
    {
        { (float)Math.Cos(angle_y), 0f, -(float)Math.Sin(angle_y) },
        { 0f, 1f, 0f },
        { (float)Math.Sin(angle_y), 0f, (float)Math.Cos(angle_y) },
    };
    1 reference
    private float[,] Get_rotation_z() => new float[,]
    {
        { (float)Math.Cos(angle_z + add_yaw), -(float)Math.Sin(angle_z + add_yaw), 0f },
        { (float)Math.Sin(angle_z + add_yaw),  (float)Math.Cos(angle_z + add_yaw), 0f },
        { 0f, 0f,  1f},
    };
    3 references
    private float[,] Multiplication(float[,] vec_1, float[,] vec_2)
    {
        float[,] Result = new float[vec_1.GetLength(0), vec_2.GetLength(1)];

        for (int i = 0; i < vec_1.GetLength(0); i++)
            for (int j = 0; j < vec_2.GetLength(1); j++)
                for (int k = 0; k < vec_2.GetLength(0); k++)
                    Result[i, j] += vec_1[i, k] * vec_2[k, j];
        return Result;
    }
```

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 Мосар систем. Исходные файлы проекта находятс на Github - https://github.com/petayyyy/OpenGl_Sharp/tree/master/Ing_progect_6_sem.

Приложение А

Листинг А-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;
        }
    }
}
```