МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ



Кафедра СМАРТ технологий

Лабораторная работа № 5

«Применение технологий дополненной реальности»

По дисциплине «Технологии визуализации данных систем управления»

Группа: 221-327

Студент: Шурова Д. С.

Дата: 04.06.2025

Преподаватель: Идиатуллов Т. Т.

Москва, 2025

**Задача**

* Разработать приложение для визуализации технической информации по роботизированной производственной линии.
* Коммуникацию по сбору данных с производственного оборудования обеспечить с использованием платформы ThingWorx, иной технологии сбора данных или посредством непосредственного подключения к оборудованию.
* Реализовать детекцию положения системы координат и визуализацию данных средствами технологий дополненной реальности.
* Выполнить разработку сцены дополненной реальности в соответствии индивидуальным заданием.

**Ход работы**

Реализовано приложение с возможность подключения к симулятору IoT и получения с него данных по протоколу Udp. Данные выводятся в listBox, а значения хранятся в глобальных переменных (Рисунок 1).

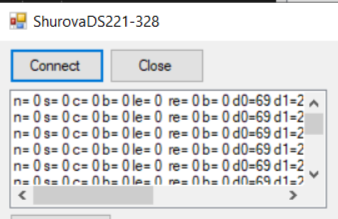


Рисунок 1 – Подключение к симулятору

Реализован вывод видео в PictureBox и с помощью библиотек OpenCvSharp, SharpGL, AForge.NET определяется положение ARuCo маркера, его координат. Относительно координат маркера строится виртуальная плоскость на которую выводится сформированная картинка в BitMap, на которую выводится текст с значениями полученными от симулятора (Рисунок 2).

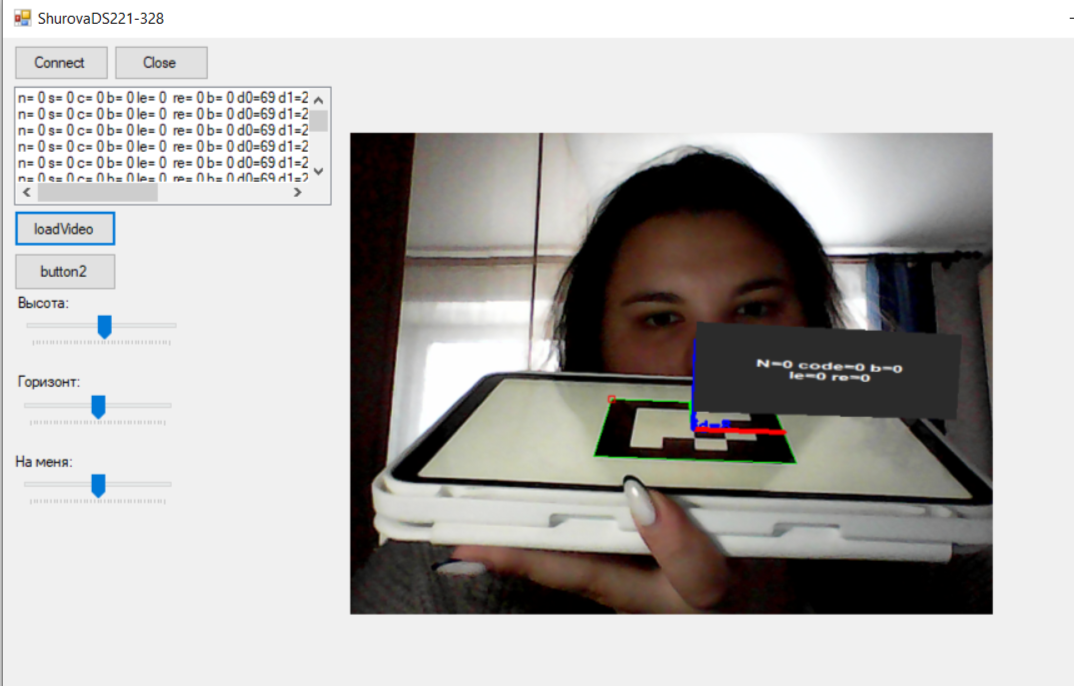


Рисунок 2 – Считывание ARuCo маркера и вывод значений с симулятора

Также в интерфейс добавлены trackBar’ы с помощью которых происходит сдвиг плоскости относительно координат маркера (Рисунок 3).

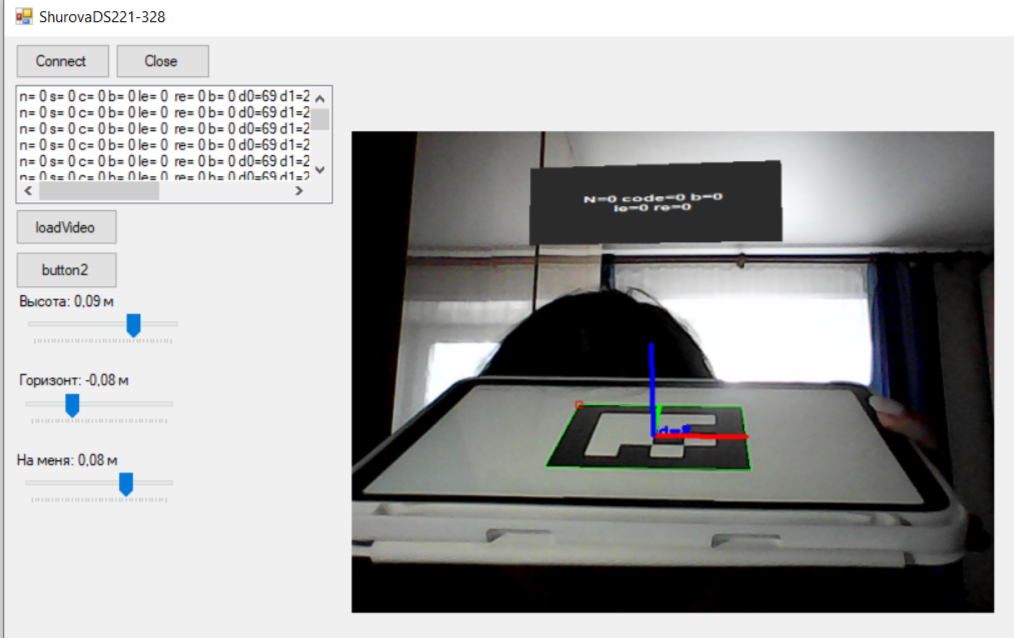


Рисунок 3 – Сдвиг виртуальное плоскость

**Вывод**

Было реализовано приложение для приема данных с симулятора по средствам протокола Udp. Вывод видео и считывания с него ARuCo маркера, относительно координат которого выводятся значения с симулятора, создание дополнительной реальности.

**Приложение А**

Листинг 1 – Код программы

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Net;

using System.Net.Sockets;

using System.Threading;

using System.Timers;

using System.Runtime.InteropServices.ComTypes;

using System.Text.RegularExpressions;

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

using System.Diagnostics;

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

using System.Xml.Linq;

using OpenCvSharp;

using OpenCvSharp.Aruco;

using System.IO;

using OpenCvSharp.Extensions;

using System.Drawing;

namespace Laba2

{

public partial class Form1 : Form

{

public delegate void ShowMessage(string message);

public ShowMessage myDelegate;

UdpClient udpClient;

Thread thread;

bool isClientOpen = false;

public int n, s;

public int le;

public int re;

public int b;

public int az;

public int d0, d1, d2, d3, d4, d5, d6, d7;

int c;

private VideoCapture capture;

private Mat frame = new Mat();

Dictionary ff = CvAruco.GetPredefinedDictionary(PredefinedDictionaryName.Dict4X4\_50);

private readonly Mat cameraMatrix;

private readonly Mat distCoeffs;

private Mat \_virtualFrame = new Mat();

public Form1()

{

InitializeComponent();

txtStartIP.Text = "127.0.0.1";

// Инициализация матрицы камеры

cameraMatrix = new Mat(3, 3, MatType.CV\_32FC1);

cameraMatrix.Set(0, 0, 1.35662728e+03f);

cameraMatrix.Set(0, 1, 0f);

cameraMatrix.Set(0, 2, 2.91998600e+02f);

cameraMatrix.Set(1, 0, 0f);

cameraMatrix.Set(1, 1, 1.37532524e+03f);

cameraMatrix.Set(1, 2, 2.25387379e+02f);

cameraMatrix.Set(2, 0, 0f);

cameraMatrix.Set(2, 1, 0f);

cameraMatrix.Set(2, 2, 1f);

// Инициализация коэффициентов дисторсии

distCoeffs = new Mat(14, 1, MatType.CV\_32FC1);

distCoeffs.Set(0, 0, -1.32575155e+00f);

distCoeffs.Set(1, 0, -7.35188200e+00f);

distCoeffs.Set(2, 0, 4.29782934e-02f);

distCoeffs.Set(3, 0, 7.66436446e-02f);

distCoeffs.Set(4, 0, 5.18928027e+01f);

}

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

{

myDelegate = new ShowMessage(ShowMessageMethod);

}

private void Form1\_KeyDown(object sender, KeyEventArgs e)

{

if(e.KeyCode == Keys.Escape)

{

thread.Abort();

udpClient.Close();

Close();

}

}

private void ReceiveMessage()

{

if (isClientOpen)

{

while (true)

{

IPEndPoint remoteIPEndpoint = new IPEndPoint(IPAddress.Any, Int32.Parse(myPort.Text));

byte[] content = udpClient.Receive(ref remoteIPEndpoint);

if (content.Length > 0)

{

string message = Encoding.ASCII.GetString(content);

this.Invoke(myDelegate, new object[] { message });

if (message.Length > 50)

{

// При выборе новой строки в ListBox вызываем обработку строки

string selectedString = message;

DataRecieve(selectedString);

}

}

}

}

}

public void DataRecieve(string message)

{

string[] dataEx = Regex.Split(message, @"\D+");

n = int.Parse(dataEx[1]);

s = int.Parse(dataEx[2]);

c = int.Parse(dataEx[3]);

le = int.Parse(dataEx[4]);

re = int.Parse(dataEx[5]);

az = int.Parse(dataEx[6]);

b = int.Parse(dataEx[7]);

d0 = int.Parse(dataEx[9]);

d1 = int.Parse(dataEx[11]);

d2 = int.Parse(dataEx[13]);

d3 = int.Parse(dataEx[15]);

d4 = int.Parse(dataEx[17]);

d5 = int.Parse(dataEx[19]);

d6 = int.Parse(dataEx[21]);

d7 = int.Parse(dataEx[23]);

//return new[] { n, s, c, le, re, az, b, d0, d1, d2, d3, d4, d5, d6, d7, l0, l1, l2, l3, l4 };

/\*dataR = new Dictionary<string, int>

{

{ "n", n },

{ "le", le },

{ "re", re },

{ "b", b },

{ "l0", l0 },

{ "l1", l1 },

{ "l2", l2 },

{ "l3", l3 },

{ "l4", l4 },

};\*/

}

private void ShowMessageMethod (string message)

{

richText.Items.Insert(0, $"n= {n} s= {s} c= {c} b= {b} le= {le} re= {re} b= {b} d0={d0} d1={d1} d2={d2} d3={d3} d4={d4} d5={d5} d6={d6} d7={d7} ");

}

private void trackBar1\_Scroll(object sender, EventArgs e)

{

labelHeight.Text = $"Высота: {trackBarHeight.Value / 100.0:F2} м";

}

private double markerOffsetZ => trackBarHeight.Value / 100.0; // в метрах, отрицательное = "вверх"

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

{

thread.Abort();

udpClient.Close();

BtnClose.Visible = false;

isClientOpen = false;

}

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

{

try

{

udpClient = new UdpClient(int.Parse(myPort.Text));

BtnClose.Visible = true;

isClientOpen = true;

thread = new Thread(new ThreadStart(ReceiveMessage));

thread.IsBackground = true;

thread.Start();

// Устанавливаем интервал в миллисекундах (в данном случае 200 мс)

myTimer.Interval = 200;

// Запускаем таймер

myTimer.Start();

}

catch

{

MessageBox.Show("Connect error");

}

}

private void SendUDPMessage(string s)

{

if (udpClient != null)

{

Int32 port = int.Parse(portClient.Text);

IPAddress ip = IPAddress.Parse(txtStartIP.Text.Trim());

IPEndPoint ipEndPoint = new IPEndPoint(ip, port);

byte[] content = Encoding.ASCII.GetBytes(s);

try

{

int count = udpClient.Send(content, content.Length, ipEndPoint);

if (count > 0)

{

ShowMessageMethod("Message has been sent.");

}

}

catch

{

ShowMessageMethod("Error occurs.");

}

}

}

private void trackBar1\_Scroll\_1(object sender, EventArgs e)

{

labelHorizent.Text = $"Горизонт: {trackBar1.Value / 100.0:F2} м";

}

private double markerOffsetX => trackBar1.Value / 100.0; // в метрах, отрицательное = "вверх"

private void trackBar2\_Scroll(object sender, EventArgs e)

{

labelOnMe.Text = $"На меня: {trackBar2.Value / 100.0:F2} м";

}

private double markerOffsetY => trackBar2.Value / 100.0; // в метрах, отрицательное = "вверх"

private void SendValues(int nm, int fm, int bm)

{

//string s = MyMessage.Text;

string s = "{ \"N\":" + nm +", \"M\":0, \"F\":"+ fm +", \"B\":" + bm + ", \"T\":0}\n";

SendUDPMessage(s);

}

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

{

if (capture == null || !capture.IsOpened()) return;

capture.Read(frame);

if (frame.Empty())

{

timer1.Stop();

return;

}

string displayText = $"N={n} code={c} b={b}\nle={le} re={re}"; // или любое другое содержимое

Bitmap bmp = CreateTextBitmap(displayText);

\_virtualFrame = BitmapConverter.ToMat(bmp); // обновляем виртуальную плоскость

// Обработка кадра с ArUco

Mat processedFrame = DetectMarkers(frame);

// Отображаем в PictureBox

videoBox.Image?.Dispose();

videoBox.Image = BitmapConverter.ToBitmap(processedFrame);

}

private void loadVideo\_Btn\_Click(object sender, EventArgs e)

{

OpenFileDialog openFileDialog = new OpenFileDialog();

openFileDialog.Filter = "Video Files|\*.mp4;\*.avi|All Files|\*.\*";

if (openFileDialog.ShowDialog() == DialogResult.OK)

{

capture = new VideoCapture(openFileDialog.FileName);

}

else

{

capture = new VideoCapture(0); // камера

}

if (capture.IsOpened())

{

timer1.Start();

}

else

{

MessageBox.Show("Не удалось открыть видео или камеру.");

}

}

// Список для хранения предыдущих позиций tvec (для сглаживания)

private List<Point3f> previousTvecs = new List<Point3f>();

private const float smoothingFactor = 0.5f; // Коэффициент сглаживания

private Mat DetectMarkers(Mat work\_flow)

{

Mat cam\_matrix = new Mat(3, 3, MatType.CV\_32FC1);

cam\_matrix.Set(0, 0, 1356.6f);

cam\_matrix.Set(0, 2, 291.9f);

cam\_matrix.Set(1, 1, 1375.3f);

cam\_matrix.Set(1, 2, 225.3f);

cam\_matrix.Set(2, 2, 1.0f);

Mat dis\_coef = new Mat(1, 5, MatType.CV\_32FC1);

dis\_coef.Set(0, 0, -1.32f);

dis\_coef.Set(0, 1, -7.35f);

dis\_coef.Set(0, 2, 0.042f);

dis\_coef.Set(0, 3, 0.076f);

dis\_coef.Set(0, 4, 51.89f);

var detectorParams = new DetectorParameters();

detectorParams.CornerRefinementMethod = CornerRefineMethod.Subpix;

CvAruco.DetectMarkers(work\_flow, ff, out Point2f[][] corners, out int[] ids, detectorParams, out \_);

Mat output = work\_flow.Clone();

if (ids.Length > 0)

{

CvAruco.DrawDetectedMarkers(output, corners, ids);

Mat rvecs = new Mat();

Mat tvecs = new Mat();

CvAruco.EstimatePoseSingleMarkers(corners, 0.1F, cam\_matrix, dis\_coef, rvecs, tvecs);

for (int i = 0; i < ids.Length; i++)

{

var rvec = rvecs.Row(i);

var tvec = tvecs.Row(i);

// Сглаживание трансляции

Vec3d t = tvec.At<Vec3d>(0);

if (previousTvecs.Count > i)

{

Point3f prev = previousTvecs[i];

t = new Vec3d(

smoothingFactor \* t[0] + (1 - smoothingFactor) \* prev.X,

smoothingFactor \* t[1] + (1 - smoothingFactor) \* prev.Y,

smoothingFactor \* t[2] + (1 - smoothingFactor) \* prev.Z);

previousTvecs[i] = new Point3f((float)t[0], (float)t[1], (float)t[2]);

}

else

{

previousTvecs.Add(new Point3f((float)t[0], (float)t[1], (float)t[2]));

}

Mat tvecSmoothed = new Mat(1, 3, MatType.CV\_64FC1);

tvecSmoothed.Set(0, 0, t[0]);

tvecSmoothed.Set(0, 1, t[1]);

tvecSmoothed.Set(0, 2, t[2]);

Cv2.DrawFrameAxes(output, cam\_matrix, dis\_coef, rvec, tvecSmoothed, 0.1f \* 0.5f);

DrawVirtualScreen(output, rvec, tvecSmoothed, cam\_matrix, dis\_coef);

}

}

return output;

}

private void DrawVirtualScreen(Mat image, Mat rvec, Mat tvec, Mat cameraMatrix, Mat distCoeffs)

{

float w = 0.15f;

float h = 0.05f;

float dx = (float)markerOffsetX;

float dy = (float)markerOffsetY;

float dz = (float)markerOffsetZ;

Point3f[] screenCorners = new Point3f[]

{

new Point3f(dx, dy, dz),

new Point3f(dx + w, dy, dz),

new Point3f(dx + w, dy, dz + h),

new Point3f(dx, dy, dz + h)

};

Point2f[] projectedPoints;

using (var objectPoints = InputArray.Create(screenCorners))

using (var imagePoints = new Mat())

{

Cv2.ProjectPoints(

objectPoints,

rvec,

tvec,

cameraMatrix,

distCoeffs,

imagePoints

);

imagePoints.GetArray<Point2f>(out projectedPoints);

}

if (\_virtualFrame.Empty()) return;

// Преобразуем в цветное изображение, если нужно

Mat virtualFrameColor = new Mat();

if (\_virtualFrame.Channels() == 1)

Cv2.CvtColor(\_virtualFrame, virtualFrameColor, ColorConversionCodes.GRAY2BGR);

else

virtualFrameColor = \_virtualFrame.Clone();

// Получаем перспективное преобразование

Point2f[] srcCorners =

{

new Point2f(0, 0),

new Point2f(virtualFrameColor.Cols - 1, 0),

new Point2f(virtualFrameColor.Cols - 1, virtualFrameColor.Rows - 1),

new Point2f(0, virtualFrameColor.Rows - 1)

};

Mat perspectiveMatrix = Cv2.GetPerspectiveTransform(srcCorners, projectedPoints);

Mat warpedFrame = new Mat(image.Size(), image.Type());

// Преобразуем с прозрачными границами

Cv2.WarpPerspective(

virtualFrameColor,

warpedFrame,

perspectiveMatrix,

image.Size(),

InterpolationFlags.Linear,

BorderTypes.Transparent

);

// Создаем маску из преобразованного изображения

Mat mask = new Mat();

Cv2.CvtColor(warpedFrame, mask, ColorConversionCodes.BGR2GRAY);

Cv2.Threshold(mask, mask, 10, 255, ThresholdTypes.Binary);

// Наложение с маской

Mat roi = new Mat(image, new Rect(0, 0, warpedFrame.Cols, warpedFrame.Rows));

warpedFrame.CopyTo(roi, mask);

}

private Bitmap CreateTextBitmap(string text, int width = 220, int height = 150)

{

Bitmap bmp = new Bitmap(width, height, System.Drawing.Imaging.PixelFormat.Format24bppRgb);

using (Graphics g = Graphics.FromImage(bmp))

{

// Переворачиваем систему координат по вертикали

g.ScaleTransform(1, -1); // Масштабирование по Y с коэффициентом -1

g.TranslateTransform(0, -height); // Сдвигаем начало координат обратно в верхний левый угол

// Заливаем темно-серым фоном с небольшой прозрачностью

g.Clear(Color.FromArgb(220, 50, 50, 50));

using (Font font = new Font("Arial", 12, FontStyle.Bold))

using (SolidBrush brush = new SolidBrush(Color.White))

{

StringFormat format = new StringFormat()

{

Alignment = StringAlignment.Center,

LineAlignment = StringAlignment.Center

};

RectangleF rect = new RectangleF(5, 5, width - 10, height - 10);

g.DrawString(text, font, brush, rect, format);

}

}

return bmp;

}

}

}

//{"N":1, "M":0, "F":50, "B":10, "T":0}