МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ автономное ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ



КУРСОВОЙ ПРОЕКТ:

«Визуализация мониторинговых данных функционирования промышленного оборудования средствами дополненной реальности»

По дисциплине: «Технологии визуализации данных системы управления»

Группа 221-327

Студент Шурова Д.С.

Дата 30.05.2025

Преподаватель Логунова Е.А.

2025

**Содержание**

[Цель работы 3](#_Toc199495951)

[Ход работы 4](#_Toc199495952)

[Вывод 8](#_Toc199495953)

[Приложение А 9](#_Toc199495954)

# **Цель работы**

Разработать систему, отображающую параметры работы промышленного робота-манипулятора, в том числе планируемую траекторию движения, с использованием средств дополненной реальности.

**Вариант 1.1**

* Определить пространственную схему размещения визуальных элементов (виджетов) для отображения мониторинговых данных;
* Выполнить покадровое считывание методами openCV или AForgeNET;
* Распознать маркер на изображении, вычислить положение системы координат производственной ячейки относительно изображения (камеры);
* 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);
* Реализовать нанесение мониторинговых данных поверх изображения с учетом схемы размещения виджетов;
* Реализовать отображение планируемой траектории движения в координатной системе, связанной с опорой робота;
* Реализовать показ изображений и сохранения их в виде видеопоследовательности (видеофайла).

# **Ход работы**

Приложение разработано на C# с использованием библиотек OpenCvSharp для обработки изображений и Windows Forms для создания графического интерфейса (Рисунок 1). Основные компоненты системы:

* Модуль захвата и обработки видеопотока;
* Детектор ArUco-маркеров;
* Калькулятор позиции робота в 3D-пространстве;
* Визуализатор траектории движения;
* Модуль записи результирующего видео.

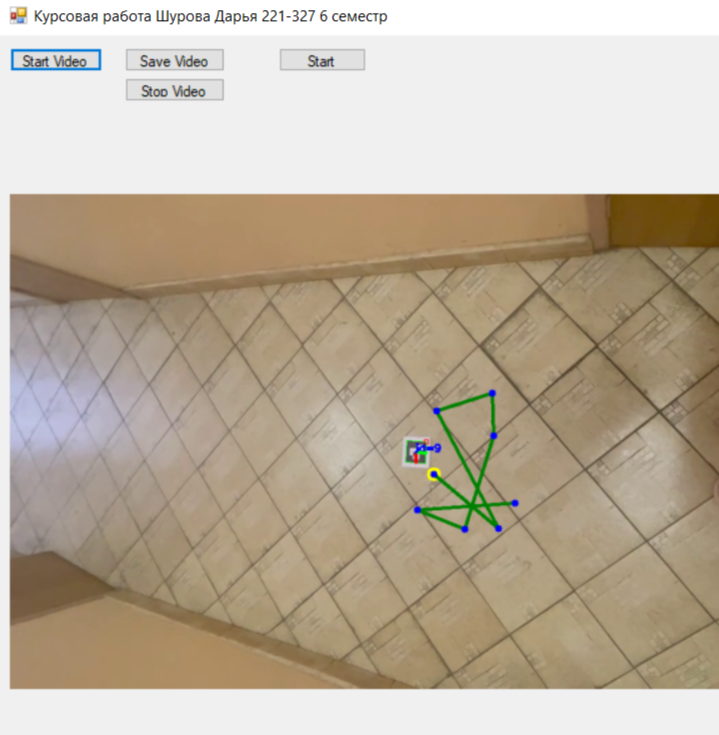
****

Рисунок 1 – Интерфейс программы

Визуализация включает следующие элементы:

* Желтая точка: текущее положение робота в пространстве;
* Синие точки: узловые точки траектории движения;
* Зеленая линия: соединение точек траектории;
* Красная точка: анимированная текущая цель движения;
* Система координат: оси XYZ, отображающие ориентацию робота.

Элементы размещаются непосредственно на видеоизображении с учетом перспективы и положения камеры, создавая эффект дополненной реальности (Рисунок 2).

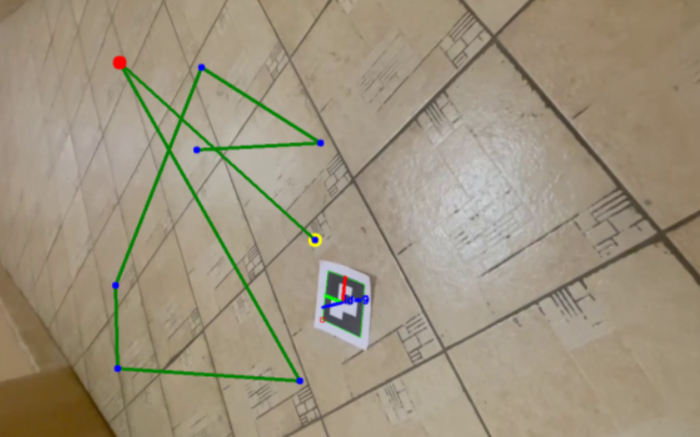


Рисунок 2 – Визуализация траектории

Система использует класс VideoCapture из OpenCvSharp для чтения видеофайлов форматов MP4 и AVI. В обработчике таймера (timer\_Tick) реализовано (Рисунок 3):

* Последовательное считывание кадров;
* Проверка окончания видео;
* Передача кадра в конвейер обработки;
* Отображение результата в интерфейсе;
* Запись в выходной файл при активации записи.

Для определения позиции робота используется следующие ключевые параметры:

* Словарь маркеров: Dict4X4\_50 (4x4 бит, 50 уникальных маркеров);
* Размер маркера: 0.1 метра;
* Матрица камеры и коэффициенты дисторсии предварительно калиброваны.

Траектория движения загружается из CSV-файла, связанного с видеофайлом. Формат данных:

1. Первая строка: абсолютные координаты робота (X,Y,Z);
2. Последующие строки: относительные координаты точек траектории.

Процесс отображения траектории включает:

1. Преобразование точек траектории в мировые координаты относительно робота;
2. Проецирование 3D-точек на 2D-плоскость изображения;
3. Отрисовку элементов.

Для визуализации движения по траектории реализована анимация:

* Перемещение красной точки по узлам траектории;
* Контроль скорости анимации через параметр framesPerPoint;
* Циклическое воспроизведение траектории.

Система обеспечивает запись результирующего видео с наложенной траекторией по нажатию кнопки старта записи видео и прекращается при нажатии стопа записи видео (Рисунок 3).

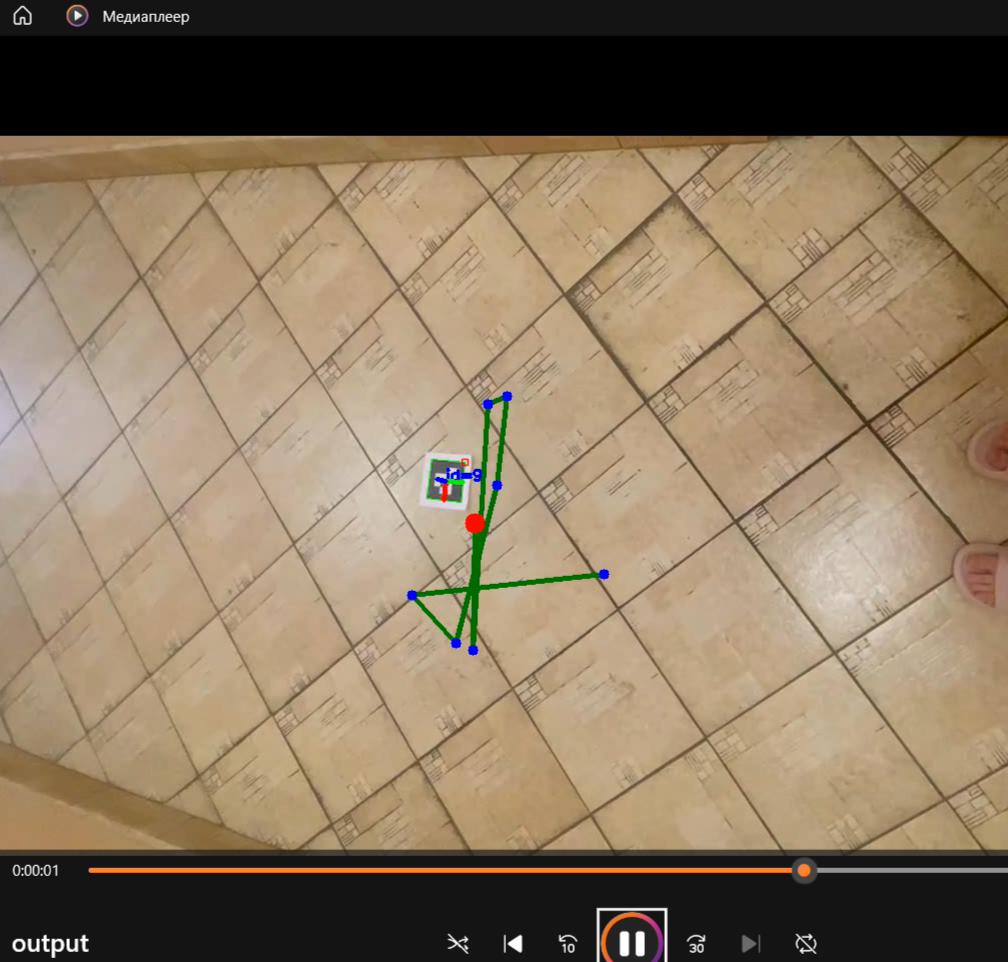


Рисунок 3 – Записанное видео

# **Вывод**

Разработано приложение в C#, которое по загруженной карте, строит маршрут для движения робота, после чего начинается отправлять команды для симуляции.

# **Приложение А**

**Программный код**

using System;

using System.Collections.Generic;

using System.IO;

using System.Windows.Forms;

using OpenCvSharp;

using OpenCvSharp.Aruco;

using OpenCvSharp.Extensions;

using System.Globalization;

namespace CourseWorkTV

{

public partial class Form1 : Form

{

private VideoCapture capture;

private Mat frame = new Mat();

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

private readonly Mat cameraMatrix;

private readonly Mat distCoeffs;

private Point3f robotPosition;

private Mat robotRvec; // Вектор вращения робота

private Mat robotTvec; // Вектор позиции робота

private Point3f[] trajectoryPoints;

private int currentPointIndex = -1;

private bool animationStarted = false;

private int frameCounter = 0;

private const int framesPerPoint = 10;

private VideoWriter videoWriter;

private bool isRecording = false;

public Form1()

{

InitializeComponent();

// 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);

// Остальные коэффициенты (нули)

for (int i = 5; i < 14; i++)

{

distCoeffs.Set(i, 0, 0f);

}

}

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);

// Загружаем CSV по тому же пути

LoadTrajectoryData(openFileDialog.FileName);

}

if (capture.IsOpened()) { timer.Start(); }

}

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

{

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

capture.Read(frame);

if (frame.Empty())

{

timer.Stop();

return;

}

// Обработка кадра с обнаружением маркера

Mat processedFrame = ProcessFrame(frame);

videoBox.Image = processedFrame.ToBitmap();

// Управление анимацией

if (animationStarted && trajectoryPoints != null && trajectoryPoints.Length > 0)

{

frameCounter++;

if (frameCounter >= framesPerPoint)

{

frameCounter = 0;

currentPointIndex = (currentPointIndex + 1) % trajectoryPoints.Length;

}

}

// Записываем ОБРАБОТАННЫЙ кадр с траекторией

if (isRecording && videoWriter.IsOpened())

{

videoWriter.Write(processedFrame); // Важно: используем processedFrame вместо frame!

}

}

private Mat ProcessFrame(Mat inputFrame)

{

Mat outputFrame = inputFrame.Clone();

var detectorParameters = new DetectorParameters();

detectorParameters.CornerRefinementMethod = CornerRefineMethod.Subpix;

CvAruco.DetectMarkers(

outputFrame,

ff,

out Point2f[][] corners,

out int[] ids,

detectorParameters,

out \_

);

if (ids != null && ids.Length > 0)

{

float markerSize = 0.1f;

Mat rvec = new Mat();

Mat tvec = new Mat();

CvAruco.EstimatePoseSingleMarkers(

corners,

markerSize,

cameraMatrix,

distCoeffs,

rvec,

tvec

);

// Сохраняем позицию робота (предполагаем, что маркер на роботе)

robotRvec = rvec.Clone();

robotTvec = tvec.Clone();

CvAruco.DrawDetectedMarkers(outputFrame, corners, ids);

Cv2.DrawFrameAxes(outputFrame, cameraMatrix, distCoeffs, rvec, tvec, markerSize \* 0.5f);

// Рисуем траекторию относительно робота

if (robotRvec != null && robotTvec != null)

{

DrawTrajectory(outputFrame, robotRvec, robotTvec);

}

}

return outputFrame;

}

private void DrawTrajectory(Mat frame, Mat arucoRvec, Mat arucoTvec)

{

if (trajectoryPoints == null || trajectoryPoints.Length == 0) return;

Mat rotationMatrix = new Mat();

Cv2.Rodrigues(arucoRvec, rotationMatrix);

Point3f[] robotPos = new Point3f[] { new Point3f(robotPosition.X, robotPosition.Y, robotPosition.Z) }; // т.к. Tvec уже — позиция робота

Mat projectedRobot = new Mat();

Cv2.ProjectPoints(

InputArray.Create(robotPos),

InputArray.Create(arucoRvec),

InputArray.Create(arucoTvec),

InputArray.Create(cameraMatrix),

InputArray.Create(distCoeffs),

projectedRobot

);

Point2f[] projectedRobotPoints;

projectedRobot.GetArray(out projectedRobotPoints);

// Рисуем жёлтую точку на изображении

Cv2.Circle(

frame,

new OpenCvSharp.Point(

(int)projectedRobotPoints[0].X,

(int)projectedRobotPoints[0].Y),

10,

Scalar.Yellow,

-1

);

// Смещаем точки траектории относительно текущей позиции маркера (робота)

Point3f[] objectPoints = new Point3f[trajectoryPoints.Length];

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

{

objectPoints[i] = new Point3f(

robotPosition.X + trajectoryPoints[i].X,

robotPosition.Y + trajectoryPoints[i].Y,

robotPosition.Z + trajectoryPoints[i].Z);

}

Mat imagePointsMat = new Mat();

Cv2.ProjectPoints(

InputArray.Create(objectPoints),

InputArray.Create(arucoRvec),

InputArray.Create(arucoTvec),

InputArray.Create(cameraMatrix),

InputArray.Create(distCoeffs),

imagePointsMat

);

Point2f[] projectedPoints;

imagePointsMat.GetArray(out projectedPoints);

// Отрисовка линии траектории

for (int i = 0; i < projectedPoints.Length - 1; i++)

{

Cv2.Line(frame,

new OpenCvSharp.Point(projectedPoints[i].X, projectedPoints[i].Y),

new OpenCvSharp.Point(projectedPoints[i + 1].X, projectedPoints[i + 1].Y),

Scalar.Green,

3

);

}

// Отрисовка точек траектории

foreach (var pt in projectedPoints)

{

Cv2.Circle(frame,

new OpenCvSharp.Point((int)pt.X, (int)pt.Y),

5,

Scalar.Blue,

-1

);

}

// 🔴 Анимация текущей точки (красная)

if (animationStarted && currentPointIndex >= 0 && currentPointIndex < projectedPoints.Length)

{

var p = projectedPoints[currentPointIndex];

Cv2.Circle(frame, new OpenCvSharp.Point((int)p.X, (int)p.Y), 10, Scalar.Red, -1);

}

}

private void LoadTrajectoryData(string videoPath)

{

string csvPath = "C:\\Users\\LENOVO\\Desktop\\proga6\\troectory.csv";

if (!File.Exists(csvPath)) return;

var lines = File.ReadAllLines(csvPath);

if (lines.Length < 2) return;

// Позиция робота (первая строка)

var robotCoords = lines[0].Split(',');

robotPosition = new Point3f(

float.Parse(robotCoords[0], CultureInfo.InvariantCulture),

float.Parse(robotCoords[1], CultureInfo.InvariantCulture),

float.Parse(robotCoords[2], CultureInfo.InvariantCulture));

trajectoryPoints = new Point3f[lines.Length - 1];

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

{

var coords = lines[i].Split(',');

trajectoryPoints[i - 1] = new Point3f(

float.Parse(coords[0], CultureInfo.InvariantCulture), // Было robotCoords[0]

float.Parse(coords[1], CultureInfo.InvariantCulture), // Было robotCoords[1]

float.Parse(coords[2], CultureInfo.InvariantCulture)); // Было robotCoords[2]

}

}

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

{

if (trajectoryPoints == null || trajectoryPoints.Length == 0)

{

MessageBox.Show("Загрузите видео с CSV-файлом траектории!");

return;

}

animationStarted = true;

currentPointIndex = 0;

frameCounter = 0;

}

Queue<Point3f> recentPositions = new Queue<Point3f>();

const int SmoothWindow = 10;

Point3f Smooth(Point3f newPos)

{

recentPositions.Enqueue(newPos);

if (recentPositions.Count > SmoothWindow)

recentPositions.Dequeue();

float sumX = 0, sumY = 0, sumZ = 0;

foreach (var p in recentPositions)

{

sumX += p.X;

sumY += p.Y;

sumZ += p.Z;

}

return new Point3f(sumX / recentPositions.Count, sumY / recentPositions.Count, sumZ / recentPositions.Count);

}

private void button\_SaveVideo\_Click(object sender, EventArgs e)

{

int width = frame.Width;

int height = frame.Height;

int fps = 30; // Частота кадров

string outputPath = "D:\\Codes\\CourseWorkTV\\output.avi";

videoWriter = new VideoWriter(

outputPath,

FourCC.MJPG, // можно заменить на FourCC.XVID

fps,

new OpenCvSharp.Size(width, height)

);

isRecording = true;

}

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

{

isRecording = false;

if (videoWriter != null)

{

videoWriter.Release();

videoWriter.Dispose();

}

}

}

}