ДОДАТОК А

Апробація результатів наукових досліджень


Primedia eLaunch
PROCEEDINGS OF THE III INTERNATIONAL SCIENTIFIC AND THEORETICAL CONFERENCE
CURRENT SCIENTIFIC GOALS, APPROACHES AND CHALLENGES
17.01.2025
RIGA
REPUBLIC OF LATVIA
SCIENTIA
COLLECTION OF SCIENTIFIC PAPERS

SECTION 12.
AUTOMATION AND APPLIANCES MAKING

Ахмад Фаді Халєд
здобувач вищої освіти факультету Автоматики і комп'ютеризованих технологій
Харківський національний університет радіоелектроніки, Україна

Науковий керівник: Новоселов Сергій Павлович
канд. техн. наук, доцент,
професор кафедри комп'ютерно-інтегрованих технологій автоматизації і робототехніки
Харківський національний університет радіоелектроніки, Україна

АВТОМАТИЗОВАНА СИСТЕМА УПРАВЛІННЯ РОБОТИЗОВАНИМ МАНІПУЛЯТОРОМ З ВИКОРИСТАННЯМ ШТУЧНОГО ІНТЕЛЕКТУ

Розробка програмного забезпечення для управління роботизованими маніпуляторами з використанням штучного інтелекту є важливим кроком у розвитку автоматизації. Завдяки використанню ШІ маніпулятори стають більш автономними, адаптивними та здатними вирішувати складні завдання в динамічних умовах. Впровадження таких технологій відкриває нові можливості для промисловості, медицини, логістики та інших галузей, де потрібна висока точність та швидкість виконання операцій.

Традиційні методи управління маніпуляторами ґрунтувалися на жорстко запрограмованих алгоритмах, які добре працювали в умовах передбачуваного середовища, але не могли адаптуватися до динамічних змін. ШІ, зокрема методи машинного навчання та глибокі нейронні мережі, дозволяють роботам вивчати нові сценарії на основі попередніх даних, робити висновки та приймати рішення самостійно. Це розширює можливості маніпуляторів, дозволяючи їм працювати в умовах неповної або неточної інформації, що важливо для багатьох сучасних виробничих процесів [1].

Основними напрямами, в яких штучний інтелект може поліпшити управління роботизованими маніпуляторами, є:

– оптимізація рухів і траєкторій;
– розпізнавання об'єктів і контроль середовища;
– адаптивне планування та прийняття рішень;
– інтерактивне навчання роботів.

Блок отримання зображення на основі веб-камери відповідає за збирання інформації про навколишнє середовище. Основним елементом для отримання зображення є веб-камера, яка захоплює зображення простору, де знаходиться об'єкт маніпуляції роботизованого маніпулятора. Отримане зображення передається до наступного блоку для обробки.

Блок обробки зображення проводить початкову обробку зображень для виділення необхідної інформації. Основним завданням є перетворення необроблених даних із камери в структури, які можуть бути проаналізовані та використані іншими блоками системи. Процеси обробки можуть включати фільтрацію шумів, сегментацію об'єктів, виявлення контурів та інші методи попередньої обробки зображень.

Центральним елементом системи є блок прийняття рішень, який координує роботу всіх інших блоків. Після отримання попередньо обробленого зображення, блок прийняття рішень визначає подальші дії системи. Цей блок управляє інформаційними потоками на основі попередньої обробки зображення засобами локального блоку OpenCV. Якщо в результаті попередньої обробки зображення не було знайдено координат об’єкту маніпуляції то зображення передається на блок аналізу зображення на основі хмарного сервісу зі штучним інтелектом.

OpenCV (Open Source Computer Vision Library) – бібліотека функцій та алгоритмів комп'ютерного зору, обробки зображень і чисельних алгоритмів загального призначення з відкритим кодом [2]. Бібліотека надає засоби для обробки і аналізу вмісту зображень, у тому числі, відстежування руху об'єктів, перетворення зображень, застосування методів машинного навчання і виявлення загальних елементів на різних зображеннях.

Після того як були отримані координати об'єкта, цей блок генерує відповідні команди для керування роботизованим маніпулятором. Він перетворює координати та параметри об'єкта в конкретні дії: переміщення маніпулятора, захоплення об'єкта тощо.

Блок управління рухами роботизованого маніпулятора відповідає за безпосереднє управління рухами маніпулятора. Це може бути реальний фізичний маніпулятор або його цифровий двійник, який моделюється для тестування та налагодження системи [1].

Роботизований маніпулятор виконує основну функцію – маніпулювання об'єктами в фізичному світі. Якщо система працює з цифровим двійником, модель маніпулятора у віртуальному середовищі допомагає проводити симуляції, прогнозувати результати і налагоджувати систему без потреби використовувати реальне обладнання.

Архітектура автоматизованої системи показана на (рис. 1).

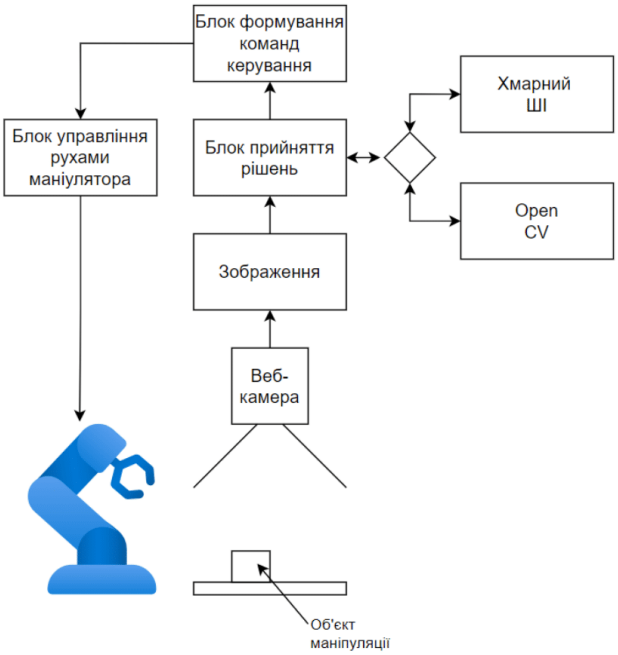

Рис. 1. **Архітектура автоматизованої системи**

Автоматизована система управління роботизованим маніпулятором складається з багатьох компонентів. Вона забезпечує обробку зображень і прийняття рішень завдяки поєднанню локальних обчислень на основі бібліотеки OpenCV та хмарного середовища, що працює з використанням технологій розпізнавання зображень зі штучним інтелектом. Кожен блок системи забезпечує роботу роботизованого маніпулятора, починаючи з отримання зображення і закінчуючи виконанням фізичних дій пристрою (маніпулятора). До складу автоматизованої системи входять:

– блок отримання зображення на основі веб-камери;

– блок обробки зображення;

– блок прийняття рішень;

– блок формування команд керування роботизованим маніпулятором;

– блок аналізу зображення на основі OpenCV;

– блок аналізу зображення на основі хмарного сервісу зі штучним інтелектом;

– блок управління рухами роботизовано го маніпулятора;

– роботизований маніпулятор, або його цифровий двійник.

Висновки. Результатом даної роботи є розроблення архітектури автоматизованої системи роботизованим маніпулятором з використанням штучного інтелекту. Описано склад розробленої системи. Вона забезпечує обробку зображень і прийняття рішень завдяки поєднанню локальних обчислень на основі бібліотеки OpenCV та хмарного середовища, що працює з використанням технологій розпізнавання зображень зі штучним інтелектом.

Список використаних джерел:

1. Невлюдов І. Ш. Застосування цифрових двійників технічних засобів автоматизації для розроблення програмно-технічних комплексів АСУ ТП : Навчальний посібник / І. Ш. Невлюдов, С. П. Новоселов, О. В. Сичова. – Харків: Видавництво Іванченка І. С., 2023. – 267 с.
2. OpenCV – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/OpenCV (дата звернення: 05.01.2025).

ДОДАТОК Б

Вихідний код програми

```
using Emgu.CV;
using Emgu.CV.CvEnum;
using Emgu.CV.Structure;
using Emgu.CV.Util;
using System.Drawing.Imaging;
using System.Runtime.InteropServices;
using System.Text.Json;
using RestSharp;

namespace WinFormsApp1
{
    public partial class Form1 : Form
    {
        string imagePath_1 = @"C:\Temp\test_image_pc.png";
        string imagePath = @"C:\Temp\clear_image_pc.png";
        string savePath = @"C:\Temp\temp_image_pc.png";
        int X_kinematic = -75;
        int y_kinematic = 100;

        private VideoCapture _capture = null; // Об'єкт для захоплення відео
        private bool _captureInProgress; // Статус захоплення відео


        public Form1()
        {
            InitializeComponent();
        }
```

```
        private void Form1_Load(object sender, EventArgs e)
        {
            pictureBox1.Image = Image.FromFile(imagePath);
            label1.Text = "";
            label2.Text = "";
        }

        private void button1_Click(object sender, EventArgs e)
        {

            if (System.IO.File.Exists(imagePath))
            {
                // Завантажуємо зображення в PictureBox
                pictureBox1.Image = Image.FromFile(imagePath);
            }
            else
            {
                MessageBox.Show("Файл зображення не знайдений!");
            }
        }

        private void button2_Click(object sender, EventArgs e)
        {
            pictureBox1.Invalidate();
            pictureBox1.Image.Save(savePath,
System.Drawing.Imaging.ImageFormat.Png);
            DetectCircles(savePath);
```

```
    Image tempImage = null;
    using (tempImage = Image.FromFile(savePath))
    {
        // Створюємо копію зображення, щоб звільнити файл
        drawKinematic1.imageContainer = new Bitmap(tempImage);
    }
}

private void DetectCircles(string imagePath)
{
    // Завантаження зображення за допомогою Emgu.CV
    Mat img = CvInvoke.Imread(imagePath, ImreadModes.Color);
    Mat grayImg = new Mat();

    // Конвертуємо зображення в відтінки сірого
    CvInvoke.CvtColor(img, grayImg, ColorConversion.Bgr2Gray);

    // Застосовуємо розмиття для зменшення шуму
    CvInvoke.GaussianBlur(grayImg, grayImg, new Size(9, 9), 2, 2);

    // Масштабування зображення до реального розміру
    float scaleFactor = 730.0f / pictureBox1.Width;
    int realWidth = (int)(img.Width * scaleFactor);
    int realHeight = (int)(img.Height * scaleFactor);
    CvInvoke.Resize(img, img, new Size(realWidth, realHeight));

    // Використовуємо метод Хаффа для пошуку кругів
```

```
        CircleF[] circles = CvInvoke.HoughCircles(grayImg,
HoughModes.Gradient, 1.0, 20.0, 100, 30, 10, 100);


        // Виводимо центри кругів на зображення
        foreach (var circle in circles)
        {
            Point center = new Point((int)circle.Center.X, (int)circle.Center.Y);
            int radius = (int)circle.Radius;


            // Малюємо центр кола на зображенні
            CvInvoke.Circle(img, center, radius, new MCvScalar(0, 0, 255), 3);


            // Виводимо координати центру кола та його радіус
            Console.WriteLine($"Коло знайдено! Центр: X = {center.X}, Y =
{center.Y}, Радіус = {radius}");
            label1.Text = String.Format("Об'єкт знайдено! Центр: X = {0}, Y =
{1}, Радіус = {2}",
                center.X, center.Y, radius);


            //Конвертування координат
            var (xConverted, yConverted) = ConvertCoordinates(center.Y,
center.X);
            label2.Text = String.Format("Координтаи для маніпулятора! Центр: X
= {0}, Y = {1}",
                (int)xConverted, (int)yConverted);
            X_kinematic = (int)xConverted;
            y_kinematic = (int)yConverted;


            // Виводимо центр на форму (можна додати ще точки чи іншу
інформацію)
```

```
            pictureBox1.Invalidate();
        }

        // Показуємо оброблене зображення в PictureBox
        Image<Bgr, byte> image = img.ToImage<Bgr, byte>();
        pictureBox1.Image = img.ToImage<Bgr, byte>().ToBitmap();
    }

    private void button3_Click(object sender, EventArgs e)
    {
        //drawKinematic1.animateKibematic(0, 100, 10);
        drawKinematic1.animateKibematic(X_kinematic, y_kinematic, 10);
    }

    public (double, double) ConvertCoordinates(double x, double y)
    {
        // Вхідні координати мають діапазон від 50 до 665
        double xInputMin = 50;
        double xInputMax = 665;
        double yInputMin = 50;
        double yInputMax = 665;

        // Вихідні координати мають діапазон для x від -75 до 75, а для y від
100 до 250
        double xOutputMin = -75;
        double xOutputMax = 75;
        double yOutputMin = 100;
        double yOutputMax = 250;
```

```
        // Лінійне масштабування для x
        double xOutput = (x - xInputMin) / (xInputMax - xInputMin) *
(xOutputMax - xOutputMin) + xOutputMin;

        // Лінійне масштабування для y
        double yOutput = (y - yInputMin) / (yInputMax - yInputMin) *
(yOutputMax - yOutputMin) + yOutputMin;

        // Повертаємо результат
        return (xOutput, yOutput);
    }

    private void pictureBox1_MouseClick(object sender, MouseEventArgs e)
    {
        if (e.Button == MouseButtons.Left)
        {
            // Викликаємо функцію для малювання кола в точці кліку
            DrawCircle(e.X, e.Y);
            //pictureBox1.Invalidate();  // Це викликає подію Paint для
перемалювання
        }
    }

    // Функція для малювання червоного кола на панелі
    private void DrawCircle(int x, int y)
    {
        // Діаметр кола
        int diameter = 60;
```

```
            // Радіус кола
            int radius = diameter / 2;

            // Створюємо об'єкт Graphics для малювання на зображенні
            using (Graphics g = Graphics.FromImage(pictureBox1.Image))
            {
                // Встановлюємо колір кола (червоний)
                using (Brush brush = new SolidBrush(Color.Red))
                {
                    // Малюємо зафарбоване коло, центр якого - точка кліку (x, y)
                    g.FillEllipse(brush, x - radius, y - radius, diameter, diameter);
                }
            }
            pictureBox1.Invalidate();
        }

        private void ClearDrawings()
        {
            if (pictureBox1.Image != null)
            {
                // Створюємо новий Bitmap з тими ж розмірами, що й зображення PictureBox
                Bitmap bmp = new Bitmap(pictureBox1.Image.Width, pictureBox1.Image.Height);

                // Заповнюємо його білим кольором (або будь-яким іншим фоном)
                using (Graphics g = Graphics.FromImage(bmp))
                {
                    g.Clear(Color.White);
```

```
            }

            // Встановлюємо нове, очищене зображення в PictureBox
            pictureBox1.Image = bmp;
            pictureBox1.Invalidate(); // Оновлюємо PictureBox
        }
    }

    private void button4_Click(object sender, EventArgs e)
    {
        if (_capture == null)
        {
            try
            {
                // Ініціалізуємо захоплення відео з веб-камери
                _capture = new VideoCapture();
                _capture.ImageGrabbed += ProcessFrame; // Додаємо обробник для
кожного кадру
            }
            catch (Exception ex)
            {
                MessageBox.Show("Помилка доступу до камери: " + ex.Message);
            }
        }

        if (_capture != null)
        {
            if (_captureInProgress)
```

```
        {
            // Якщо відео вже йде, зупиняємо його
            _capture.Pause();
            button4.Text = "Старт";
        }
        else
        {
            // Запускаємо відео
            _capture.Start();
            button4.Text = "Пауза";
        }

        _captureInProgress = !_captureInProgress;
    }
}

private void button5_Click(object sender, EventArgs e)
{
    if (_capture != null)
    {
        _capture.Stop(); // Зупиняємо захоплення
        pictureBox1.Image = null; // Очищуємо зображення
        button4.Text = "Start";
        _captureInProgress = false;
    }
}

// Обробник захоплення кожного кадру з відео
```

```
    private void ProcessFrame(object sender, EventArgs arg)
    {
        Mat frame = new Mat();
        _capture.Retrieve(frame, 0); // Захоплюємо кадр

        // Використовуємо Invoke для безпечного оновлення UI
        if (pictureBox1.InvokeRequired)
        {
            pictureBox1.Invoke(new Action(() => pictureBox1.Image =
frame.ToBitmap()));
        }
        else
        {
            pictureBox1.Image = frame.ToBitmap();
        }
    }

    private void Form1_FormClosed(object sender, FormClosedEventArgs e)
    {
        if (_capture != null)
        {
            _capture.Dispose(); // Звільняємо ресурс захоплення відео
        }
    }

    static string SendOpenAIRequest(string apiKey, string base64Image)
    {
```

```
// URL API OpenAI
string apiUrl = "https://api.openai.com/v1/chat/completions";

// Створення клієнта для відправки запиту
var client = new RestClient(apiUrl);

// Створення запиту
var request = new RestRequest()
    .AddHeader("Authorization", $"Bearer {apiKey}")
    .AddHeader("Content-Type", "application/json");

// Формування запиту у форматі JSON
var payload = new
{
    model = "gpt-4o-mini", //"gpt-4o",
    messages = new[]
    {
    new
    {
        role = "user",
        content = new object[]
        {
            new
            {
                type = "text",
                text = "Find the coordinates of the round red object in the image.
" +
                "Provide the information in the form: " +
```

```
                "center: x=...; y=..."
            },
            new
            {
                type = "image_url",
                image_url = new
                {
                    url = $"{base64Image}"
                }
            }
        }
    },
    max_tokens = 300
};

// Серіалізація запиту у JSON
string jsonPayload = JsonSerializer.Serialize(payload);

// Додаємо тіло запиту у JSON форматі
request.AddStringBody(jsonPayload, DataFormat.Json);

// Виконуємо запит
RestResponse response = client.Post(request);

// Повертаємо відповідь
return response.Content;
}   }
```

ДОДАТОК В

Демонстраційний Матеріал

№	Позначення	Найменування	Дод. відомості
		Текстові документи	
1	*ГЮИК. 503300.002 ПЗ*	*Пояснювальна записка*	*76 с., ф. А4*
		Інші документи	
2		*Апробація результатів наукових досліджень*	*А4, 5с*
3		*Вихідний код програми*	*А4, 12с*
4		*Демонстраційний матеріал*	*А4, 12с*

					ГЮИК. 503300.002 ВД	
Змін.	Арк.	№ документа	Підп.	Дата		
Розроб.		Ахмад Ф.Х.			Розроблення програмного засобу управління роботизованим маніпулятором з використанням штучного інтелекту	Літ. Аркуш Аркушів
Перевір.		Сичова О.В.				Н 1 1
Н. контр.		Стародубцев М.Г.				ХНУРЕ Кафедра КІТАР
Затвер.		Невлюдов І.Ш				