Київський політехнічний інститут імені Ігоря Сікорського

Приладобудівний факультет

Кафедра автоматизації та систем неруйнівного контролю

Лабораторна робота №3

з предмету "Комп'ютерний зір"

Студент: Погорєлов Богдан
Група: ПК-51мп

2025 рік

Розробити програму, яка виконує такі дії з ЦВ:

1. Програма зчитує цифрове відео з зовнішньої цифрової камери. Також ця програма розпізнає об’єкти в ЦВ використовуючи аналіз контурів об’єктів. Програма повинна застосувати мінімум одну геометричну ознаку, яку визначає розробник програми. Програма формує та відображає вихідне ЦВ, яке містить контури об’єктів, описані прямокутники та текстові рядки з інформацією про розпізнані об’єкти. При цьому програма знаходить об’єкт заданої форми (трикутних, чотирикутник і т.п.) з найбільшою площею та наводить на його геометричний центр оптичну ось цифрової камери. Для цього програма подає відповідні команди на контролер поворотних платформ з двома серво-приводами.
2. У контролер поворотних платформ записується програма з лабораторної роботи № 2

Тема: Розпізнавання образів та автоматичне слідкування за об'єктом. Варіант: 12

Мета: Реалізувати алгоритм пошуку геометричної фігури (наприклад, чотирикутника) з найбільшою площею.

Реалізувати алгоритм відслідковування (Tracking): обчислювати відхилення центру фігури від центру кадру та повертати сервоприводи так, щоб об'єкт завжди був по центру.

1. Програма для контролера (Arduino) Згідно із завданням, ми використовуємо той самий код, що і в Лабораторній №2. Він приймає команди Pxx Tyy і відповідає DONE.

Важливо: Для плавного слідкування бажано прибрати delay(600) у коді Arduino, якщо рухи будуть занадто повільними, але для формального виконання завдання залишаємо код без змін.

```
#include <Servo.h>

Servo servoPan;
Servo servoTilt;

const int PIN_PAN = 9;
const int PIN_TILT = 10;

String inputString = "";
bool stringComplete = false;

void setup() {
  Serial.begin(115200);
  servoPan.attach(PIN_PAN);
  servoTilt.attach(PIN_TILT);
  servoPan.write(90);
```

```
  servoTilt.write(90);
  inputString.reserve(200);
}

void loop() {
  if (stringComplete) {
    parseCommand(inputString);
    inputString = "";
    stringComplete = false;
  }
}

void serialEvent() {
  while (Serial.available()) {
    char inChar = (char)Serial.read();
    if (inChar == '\n') stringComplete = true;
    else inputString += inChar;
  }
}

void parseCommand(String command) {
  int pIndex = command.indexOf('P');
  int tIndex = command.indexOf('T');
  if (pIndex != -1 && tIndex != -1) {
    String panVal = command.substring(pIndex + 1, tIndex);
    String tiltVal = command.substring(tIndex + 1);
    int panAngle = panVal.toInt();
    int tiltAngle = tiltVal.toInt();
    servoPan.write(constrain(panAngle, 0, 180));
    servoTilt.write(constrain(tiltAngle, 0, 180));
    delay(100);
    Serial.println("DONE");
  }
}
```

2. Програма розпізнавання та трекінгу (Python) Ця програма:

Отримує зображення.

Перетворює його в бінарне (чорно-біле) для пошуку контурів.

Знаходить контури і апроксимує їх (спрощує до ліній).

Якщо контур має 4 кути (прямокутник) — він вважається цільовим.

Знаходить найбільший прямокутник.

Обчислює його центр мас.

Коригує кути сервоприводів, щоб центр об'єкта збігся з центром кадру.

In [ ]:
```
import cv2
import numpy as np
import serial
import time
```

```python
# --- НАЛАШТУВАННЯ ---
SERIAL_PORT = 'COM3'
BAUD_RATE = 115200

# Налаштування кольору для детекції (зараз налаштовано на темні об'єкти)
# Якщо потрібно детектувати за кольором, краще використати HSV.
# Тут використаємо простий Threshold (поріг яскравості).
THRESHOLD_VALUE = 100

# Параметри сервоприводів
current_pan = 90
current_tilt = 90
PAN_LIMITS = (0, 180)
TILT_LIMITS = (0, 180)

# Поріг чутливості (Dead Zone), щоб камера не тремтіла
ERROR_THRESHOLD = 40
SERVO_STEP = 2  # На скільки градусів повертати за один раз

def send_servo_command(ser, pan, tilt):
    """Відправляє команду на Arduino без блокування."""
    if ser and ser.is_open:
        cmd = f"P{pan} T{tilt}\n"
        ser.write(cmd.encode())
        # Не чекаємо "DONE" тут, щоб не зупиняти відеопотік
        # Але час від часу треба чистити буфер вводу
        if ser.in_waiting > 0:
            ser.read_all()

def main():
    global current_pan, current_tilt

    # Підключення до Arduino
    try:
        ser = serial.Serial(SERIAL_PORT, BAUD_RATE, timeout=0.1)
        time.sleep(2)
        print(f"Connected to {SERIAL_PORT}")
    except Exception as e:
        print(f"Serial Error: {e}")
        ser = None

    cap = cv2.VideoCapture(0)

    # Зменшимо роздільну здатність для швидкодії
    cap.set(3, 640)
    cap.set(4, 480)

    screen_center_x = 320
    screen_center_y = 240

    print("Tracking started. Looking for RECTANGLES (4 corners).")
    print("Press 'q' to quit.")

    while True:
        ret, frame = cap.read()
        if not ret: break

        # 1. Попередня обробка
        # Конвертація в відтінки сірого
```

```
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        # Розмиття для зменшення шуму
        blur = cv2.GaussianBlur(gray, (5, 5), 0)
        # Бінаризація (все що темніше 100 стає чорним, світліше - білим)
        # Можна використати Canny: cv2.Canny(blur, 50, 150)
        _, thresh = cv2.threshold(blur, THRESHOLD_VALUE, 255,
                                  cv2.THRESH_BINARY_INV)

        # 2. Пошук контурів
        contours, _ = cv2.findContours(thresh, cv2.RETR_TREE,
                                       cv2.CHAIN_APPROX_SIMPLE)

        max_area = 0
        best_cnt = None
        object_center = None

        for cnt in contours:
            area = cv2.contourArea(cnt)

            # Відсіюємо дрібний шум
            if area > 1000:
                # Апроксимація контуру (спрощення форми)
                peri = cv2.arcLength(cnt, True)
                approx = cv2.approxPolyDP(cnt, 0.02 * peri, True)

                # Геометрична ознака: кількість кутів = 4 (Чотирикутник)
                if len(approx) == 4:
                    if area > max_area:
                        max_area = area
                        best_cnt = approx

        # 3. Логіка трекінгу
        if best_cnt is not None:
            # Малюємо контур
            cv2.drawContours(frame, [best_cnt], -1, (0, 255, 0), 3)

            # Описуємо прямокутник навколо об'єкта
            x, y, w, h = cv2.boundingRect(best_cnt)
            cv2.rectangle(frame, (x, y), (x + w, y + h), (255, 0, 0), 2)

            # Інформаційний текст
            cv2.putText(frame, "Target: Rectangle", (x, y - 10),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)
            cv2.putText(frame, f"Area: {int(max_area)}", (x, y + h + 20),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 1)

            # Обчислення центру мас (Moment)
            M = cv2.moments(best_cnt)
            if M["m00"] != 0:
                cX = int(M["m10"] / M["m00"])
                cY = int(M["m01"] / M["m00"])
                object_center = (cX, cY)

                # Малюємо центр та вектор до центру екрану
                cv2.circle(frame, (cX, cY), 7, (0, 0, 255), -1)
                cv2.line(frame, (320, 240), (cX, cY), (0, 255, 255), 2)

                # --- АЛГОРИТМ СЛІДКУВАННЯ (Simple Proportional Control) ---

                # Відхилення по X (для Pan)
```

```
                error_x = cX - screen_center_x
                # Відхилення по Y (для Tilt)
                error_y = cY - screen_center_y

                # Коригуємо Pan (Горизонталь)
                if abs(error_x) > ERROR_THRESHOLD:
                    # Якщо об'єкт справа (error > 0), треба зменшити кут
                    # (або збільшити, залежно від конструкції серво)
                    if error_x > 0:
                        current_pan -= SERVO_STEP
                    else:
                        current_pan += SERVO_STEP

                # Коригуємо Tilt (Вертикаль)
                if abs(error_y) > ERROR_THRESHOLD:
                    if error_y > 0: # Об'єкт нижче центру
                        current_tilt += SERVO_STEP
                    else:
                        current_tilt -= SERVO_STEP

                # Обмежуємо кути
                current_pan = np.clip(current_pan, *PAN_LIMITS)
                current_tilt = np.clip(current_tilt, *TILT_LIMITS)

                # Відправляємо команду
                send_servo_command(ser, int(current_pan), int(current_tilt))

        # Малюємо центр екрану (приціл)
        cv2.circle(frame, (320, 240), 5, (255, 0, 0), -1)

        # Відображення результату
        cv2.imshow("Lab 3 - Object Tracking", frame)
        cv2.imshow("Threshold View", thresh) # Службове вікно

        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

    if ser:
        ser.close()
    cap.release()
    cv2.destroyAllWindows()

if __name__ == "__main__":
    main()
```

Висновок

У цій роботі було створено систему активного комп'ютерного зору (Active Computer Vision).

Основні досягнення:

Розпізнавання форми: Реалізовано алгоритм аналізу контурів. Програма успішно класифікує об'єкти за кількістю вершин (у даному випадку — виділяє чотирикутники).

Визначення пріоритету: Серед усіх знайдених об'єктів програма обирає "головний" за критерієм найбільшої площі.

Візуалізація: На відео накладаються контури, обмежувальні рамки (Bounding Box), центр мас та службова інформація.

Слідкуюча система: Реалізовано замкнений контур керування. Координати об'єкта з камери перетворюються на керуючі сигнали для сервоприводів. Це дозволяє камері автоматично утримувати об'єкт у центрі поля зору, компенсуючи його переміщення.

Система демонструє базові принципи роботи промислових роботів та охоронних систем з функцією автотрекінгу.