

ДОДАТОК А
Текст програми

main.py

```
import cv2
import numpy as np
import time
import json
import os
import threading
from tkinter import Tk, Label, Button, StringVar, OptionMenu, Frame
import pygame

object_in_zone = False
recording = False
out = None
running = True
processing = False

lock = threading.Lock()

MODES = ["Motion Detection", "MeanShift", "CamShift"]
current_mode = 0
restricted_zones = [
    (50, 50, 150, 150), (490, 50, 590, 150), (50, 330, 150, 430),
    (490, 330, 590, 430), (200, 350, 350, 450), (200, 50, 300, 150)
]
weights_path = "yolov4.weights"
config_path = "yolov4.cfg"
log_file = "events_log.json"
snapshot_folder = "snapshots/"

if not os.path.exists(weights_path) or not os.path.exists(config_path):
    raise FileNotFoundError("YOLO files not found. Make sure yolov4.weights and yolov4.cfg are in the
working directory.")
```

```

net = cv2.dnn.readNet(weights_path, config_path)
layer_names = net.getLayerNames()
output_layers = [layer_names[i - 1] for i in net.getUnconnectedOutLayers()]

fourcc = cv2.VideoWriter_fourcc(*'XVID')
out = None
recording = False

log_data = []
if not os.path.exists(snapshot_folder):
    os.makedirs(snapshot_folder)

pygame.mixer.init()
pygame.mixer.music.load("signal-trevogi.wav")

# GUI
root = Tk()
root.title("Motion Tracking System")

root.geometry("400x300") # Змінюємо розміри основного вікна
root.resizable(True, True) # Дозволяємо змінювати розміри вручну

control_frame = Frame(root)
control_frame.pack(fill="both", expand=True)

control_frame.grid_rowconfigure(0, weight=2)
control_frame.grid_rowconfigure(1, weight=1)
control_frame.grid_rowconfigure(2, weight=1)
control_frame.grid_columnconfigure(0, weight=1)

mode_var = StringVar(root)
mode_var.set(MODES[current_mode])

```

```

def play_alert_sound():
    pygame.mixer.music.play()

def stop_alert_sound():
    if pygame.mixer.music.get_busy():
        pygame.mixer.music.stop()

def is_in_restricted_area(x, y, zones):
    for zone in zones:
        if zone[0] <= x <= zone[2] and zone[1] <= y <= zone[3]:
            return True
    return False

def log_event(event_type, details):
    global log_data
    event = {"timestamp": time.strftime("%Y-%m-%d %H:%M:%S"), "type": event_type, "details":
details}
    log_data.append(event)
    with open(log_file, "w") as f:
        json.dump(log_data, f, indent=4)

def switch_mode(mode):
    global current_mode
    current_mode = MODES.index(mode)
    print(f"Switched to mode: {mode}")

Label(control_frame, text="Select Mode:").grid(row=0, column=0, padx=30, pady=30, sticky="nsew")
mode_menu = OptionMenu(control_frame, mode_var, *MODES, command=switch_mode)
mode_menu.grid(row=2, column=0, padx=10, pady=10, sticky="nsew")

def start_system():

```

```

global processing
processing = True
print("[INFO] System started.")

def stop_system():
    global running, processing, out
    running = False
    processing = False
    if out:
        out.release()
    cv2.destroyAllWindows()
    root.quit()

Button(control_frame, text="Start", command=start_system).place(relx=0.1, rely=0.3, relwidth=0.8,
relheight=0.2)
Button(control_frame, text="Stop", command=stop_system).place(relx=0.1, rely=0.52, relwidth=0.8,
relheight=0.2)

def process_frame_with_yolo(frame, restricted_zones, net, output_layers, conf_threshold=0.6,
nms_threshold=0.4):

    height, width = frame.shape[:2]

    blob = cv2.dnn.blobFromImage(frame, 1 / 255.0, (416, 416), swapRB=True, crop=False)
    net.setInput(blob)
    detections = net.forward(output_layers)

    boxes = []
    confidences = []
    detected_people = []
    current_object_in_zone = False

    for output in detections:
        for detection in output:

```

```

scores = detection[5:]
class_id = np.argmax(scores)
confidence = scores[class_id]

if class_id == 0 and confidence > conf_threshold:
    box = detection[0:4] * np.array([width, height, width, height])
    (centerX, centerY, boxW, boxH) = box.astype("int")
    x = int(centerX - boxW / 2)
    y = int(centerY - boxH / 2)
    w = int(boxW)
    h = int(boxH)
    boxes.append([x, y, w, h])
    confidences.append(float(confidence))

indices = cv2.dnn.NMSBoxes(boxes, confidences, conf_threshold, nms_threshold)

if len(indices) > 0:
    for i in indices.flatten():
        x, y, w, h = boxes[i]
        detected_people.append((x, y, w, h))
        centerX, centerY = x + w // 2, y + h // 2

        for zone in restricted_zones:
            if zone[0] <= centerX <= zone[2] and zone[1] <= centerY <= zone[3]:
                current_object_in_zone = True

                cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
                cv2.putText(frame, "Person in Zone", (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,
255, 0), 2)
                break
            else:

                cv2.rectangle(frame, (x, y), (x + w, y + h), (255, 0, 0), 2)

```

```
cv2.putText(frame, "Person", (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 0, 0), 2)
```

```
for zone in restricted_zones:
```

```
    cv2.rectangle(frame, (zone[0], zone[1]), (zone[2], zone[3]), (0, 0, 255), 2)
```

```
    cv2.putText(frame, "Restricted Zone", (zone[0], zone[1] - 10), cv2.FONT_HERSHEY_SIMPLEX,
0.5, (0, 0, 255), 2)
```

```
return frame, current_object_in_zone
```

```
roi_hist = None
```

```
track_window = None
```

```
roi_selected = False
```

```
current_object_in_zone = False
```

```
def run_system():
```

```
    cap = cv2.VideoCapture(0)
```

```
    if not cap.isOpened():
```

```
        print("Error: Cannot access the camera")
```

```
        return
```

```
    cv2.namedWindow("Frame", cv2.WINDOW_NORMAL)
```

```
    cv2.resizeWindow("Frame", 800, 600)
```

```
    global object_in_zone, recording, out, prev_frame, roi_hist, track_window, roi_selected,
current_object_in_zone
```

```
    prev_frame = None
```

```
    while running:
```

```
        ret, frame = cap.read()
```

```
        if not ret:
```

```
            break
```

```
        frame = cv2.flip(frame, 1)
```

```
        key = cv2.waitKey(1)
```

```

if current_mode == 0: # Motion Detection
    gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    gray_frame = cv2.GaussianBlur(gray_frame, (21, 21), 0)

    if prev_frame is None:
        prev_frame = gray_frame
        continue

    frame_diff = cv2.absdiff(prev_frame, gray_frame)
    _, thresh = cv2.threshold(frame_diff, 25, 255, cv2.THRESH_BINARY)
    contours, _ = cv2.findContours(thresh, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)

    for contour in contours:
        if cv2.contourArea(contour) < 500:
            continue
        (x, y, w, h) = cv2.boundingRect(contour)
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

        snapshot_path = f"{snapshot_folder}motion_{time.strftime('%Y%m%d_%H%M%S')}.jpg"
        cv2.imwrite(snapshot_path, frame)
        print(f"Snapshot saved: {snapshot_path}")

        log_event("Motion Detected", {"x": x, "y": y, "w": w, "h": h})

    prev_frame = gray_frame

elif current_mode == 1:
    face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
"haarcascade_frontalface_default.xml")

    if face_cascade.empty():
        print("Error: Haar Cascade file not found.")
        break

    gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray_frame, scaleFactor=1.1, minNeighbors=5,

```

```
minSize=(50, 50))
```

```

if len(faces) > 0:
    x, y, w, h = faces[0]
    padding = 20
    x = max(0, x - padding)
    y = max(0, y - padding)
    w = min(frame.shape[1] - x, w + 2 * padding)
    h = min(frame.shape[0] - y, h + 2 * padding)

    cv2.rectangle(frame, (x, y), (x + w, y + h), (255, 0, 0), 2)
    frame_height, frame_width = frame.shape[:2]

    x = max(0, x)
    y = max(0, y)
    w = min(w, frame_width - x)
    h = min(h, frame_height - y)

    if w <= 0 or h <= 0:
        print("[ERROR] Invalid ROI size after coordinate adjustment.")
        roi_selected = False
    else:
        roi = frame[y:y + h, x:x + w]
        if roi is not None and roi.size > 0:
            roi_hsv = cv2.cvtColor(roi, cv2.COLOR_BGR2HSV)
            mask = cv2.inRange(roi_hsv, np.array((0., 60., 32.)), np.array((180., 255., 255.)))
            roi_hist = cv2.calcHist([roi_hsv], [0], mask, [180], [0, 180])
            cv2.normalize(roi_hist, roi_hist, 0, 255, cv2.NORM_MINMAX)
            track_window = (x, y, w, h)
            roi_selected = True

        else:
            print("[ERROR] ROI is None or invalid.")
            roi_selected = False
    else:
        print("[ERROR] No faces detected in MeanShift.")

```

```
cv2.putText(frame, f"Faces detected: {len(faces)}", (10, 30), cv2.FONT_HERSHEY_SIMPLEX,
0.7, (255, 0, 255), 2)
```

```
elif current_mode == 2:
```

```
if roi_selected:
```

```
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
```

```
    back_proj = cv2.calcBackProject([hsv], [0], roi_hist, [0, 180], 1)
```

```
    kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (5, 5))
```

```
    back_proj = cv2.morphologyEx(back_proj, cv2.MORPH_OPEN, kernel)
```

```
    back_proj = cv2.GaussianBlur(back_proj, (5, 5), 0)
```

```
try:
```

```
    ret, track_window = cv2.CamShift(back_proj, track_window,
                                     (cv2.TERM_CRITERIA_EPS | cv2.TERM_CRITERIA_COUNT, 10,
```

```
1))
```

```
    x, y, w, h = track_window
```

```
    frame_height, frame_width = frame.shape[:2]
```

```
    x = max(0, min(x, frame_width - 1))
```

```
    y = max(0, min(y, frame_height - 1))
```

```
    w = min(w, frame_width - x)
```

```
    h = min(h, frame_height - y)
```

```
if w <= 0 or h <= 0:
```

```
    print("[ERROR] Invalid track_window size after restriction.")
```

```
    roi_selected = False
```

```
else:
```

```
    track_window = (x, y, w, h)
```

```

        pts = cv2.boxPoints(ret)
        pts = np.array(pts, dtype=np.int32)
        cv2.polylines(frame, [pts], True, (0, 255, 255), 2)

except Exception as e:
    print(f"[ERROR] CamShift failed: {e}")

frame, current_object_in_zone = process_frame_with_yolo(
    frame, restricted_zones, net, output_layers, conf_threshold=0.6, nms_threshold=0.4
)

if current_object_in_zone and not object_in_zone:

    object_in_zone = True
    if not recording:

        recording = True
        if out is None:
            video_filename = f"{snapshot_folder}alert_{time.strftime('%Y%m%d_%H%M%S')}.avi"
            out = cv2.VideoWriter(
                video_filename,
                fourcc,
                20.0,
                (frame.shape[1], frame.shape[0])
            )
            print(f"[INFO] Video recording started: {video_filename}")

        play_alert_sound()

elif not current_object_in_zone and object_in_zone:

    object_in_zone = False
    if recording:

```

```
    recording = False
    if out is not None:
        out.release()
        out = None
        print("[INFO] Video recording stopped and saved.")

    stop_alert_sound()

    if recording and out is not None:
        out.write(frame)

    cv2.imshow("Frame", frame)

    if recording:
        if out:
            out.release()

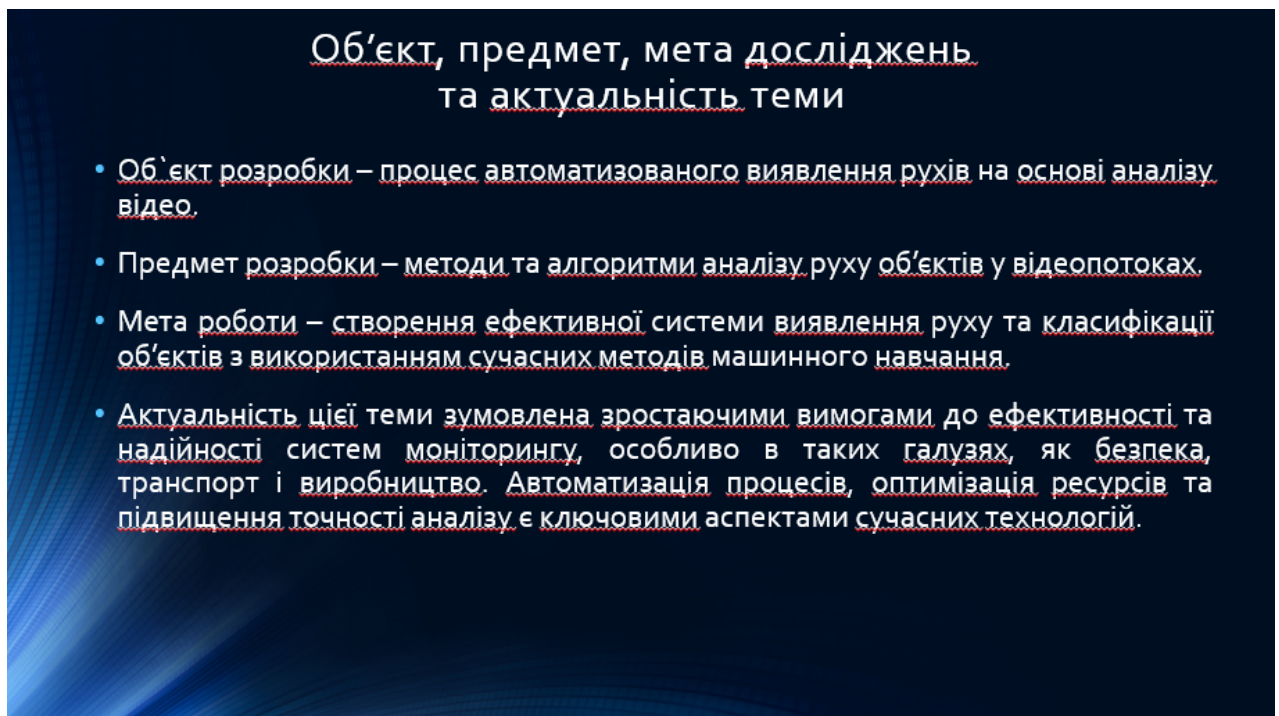
    cap.release()
    cv2.destroyAllWindows()

    threading.Thread(target=run_system, daemon=True).start()
    root.mainloop()
```

ДОДАТОК Б
Демонстраційний матеріал



РИСУНОК А.1 – ТИТУЛЬНИЙ АРККУШ

РИСУНОК А.2 – ОБ'ЄКТ, ПРЕДМЕТ, МЕТА ДОСЛІДЖЕННЯ ТА
АКТУАЛЬНІСТЬ ТЕМИ

Функціональні можливості системи для користувача

- головне меню керування;
- перегляд вікна камери в реальному часі;
- перемикання між методами обробки відео;
- запис порушення;
- відтворення звуку тривоги;



РИСУНОК А.3 – ФУНКЦІОНАЛЬНІ МОЖЛИВОСТІ СИСТЕМИ ДЛЯ КОРИСТУВАЧА

Мова програмування та середовище розробки

- Середовище розробки: PyCharm Community Edition 2024.3.
- Технічне забезпечення: ПК з процесором Core і3 і вище.
- Мови програмування: Python

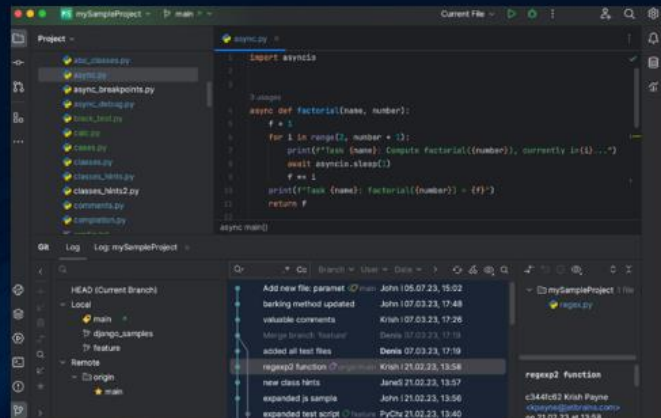


РИСУНОК А.4 – МОВА ПРОГРАМУВАННЯ ТА СЕРЕДОВИЩЕ РОЗРОБКИ

Головне меню керування

У меню керування користувачі можуть виконувати такі дії, як запуск моніторингу відео, його зупинка або перемикання між різними методами обробки відео, а саме: Motion Detection, MeanShift, CamShift. Ця функціональність забезпечує максимальну гнучкість. Інтерфейс інтуїтивно зрозумілий, що робить систему зручною для роботи навіть для користувачів без досвіду в технологіях машинного навчання.

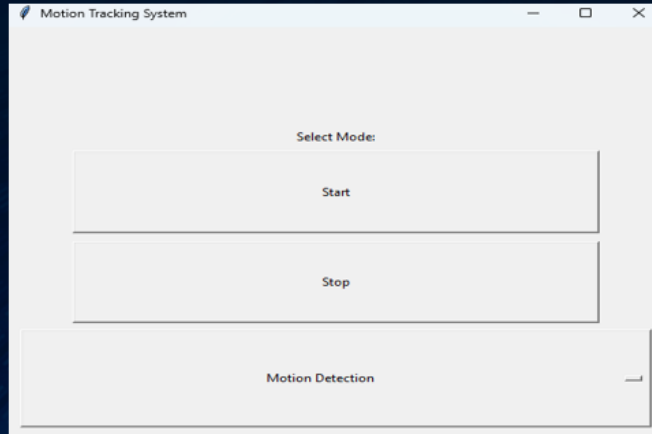


РИСУНОК А.5 – ГОЛОВНЕ МЕНЮ КЕРУВАННЯ

Метод Motion Detection

Motion Detection, або виявлення руху, базується на аналізі змін між поточними та попередніми відеокадрами для ідентифікації руху. Він спрацьовує, коли трапляються зміни відео потоці. Ці зміни виділяються квадратом та робиться фото цих змін та зберігається на комп'ютері або ноутбуці користувача з датою та часом цих змін, після зберігання фото в консоль PyCharm виводиться повідомлення про успішне збереження.

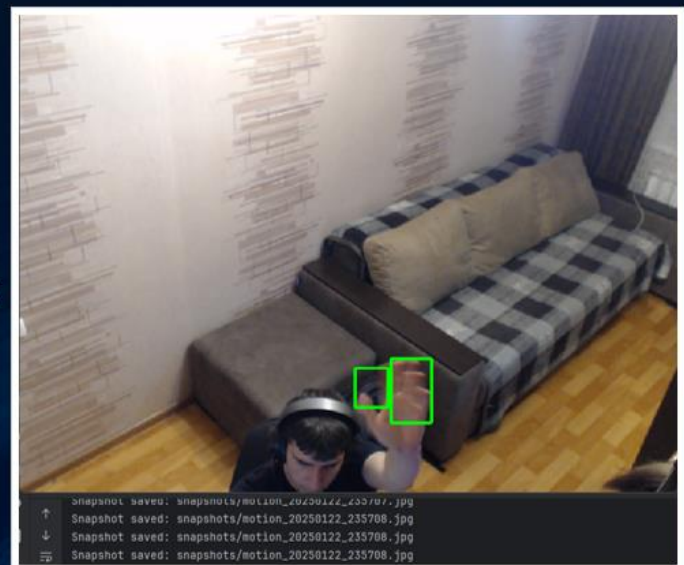


РИСУНОК А.6 – МЕТОД MOTION DETECTION

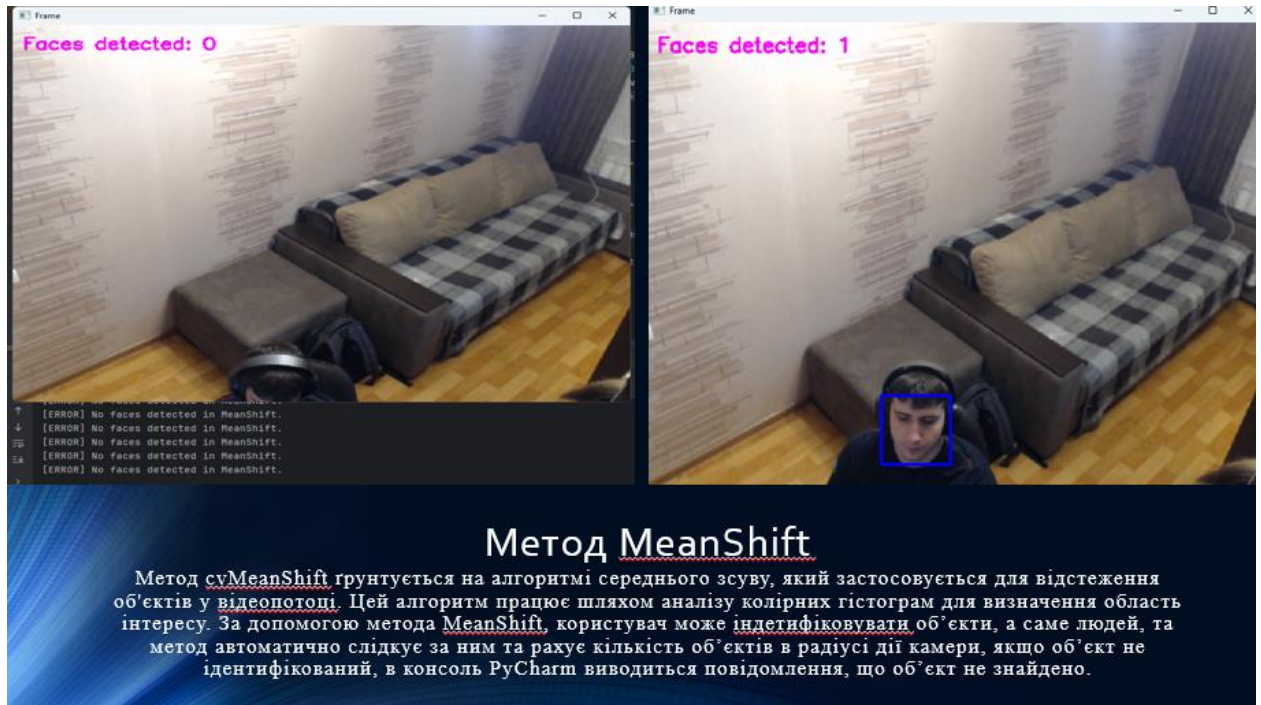


РИСУНОК А.7 – МЕТОД MEANSHIFT

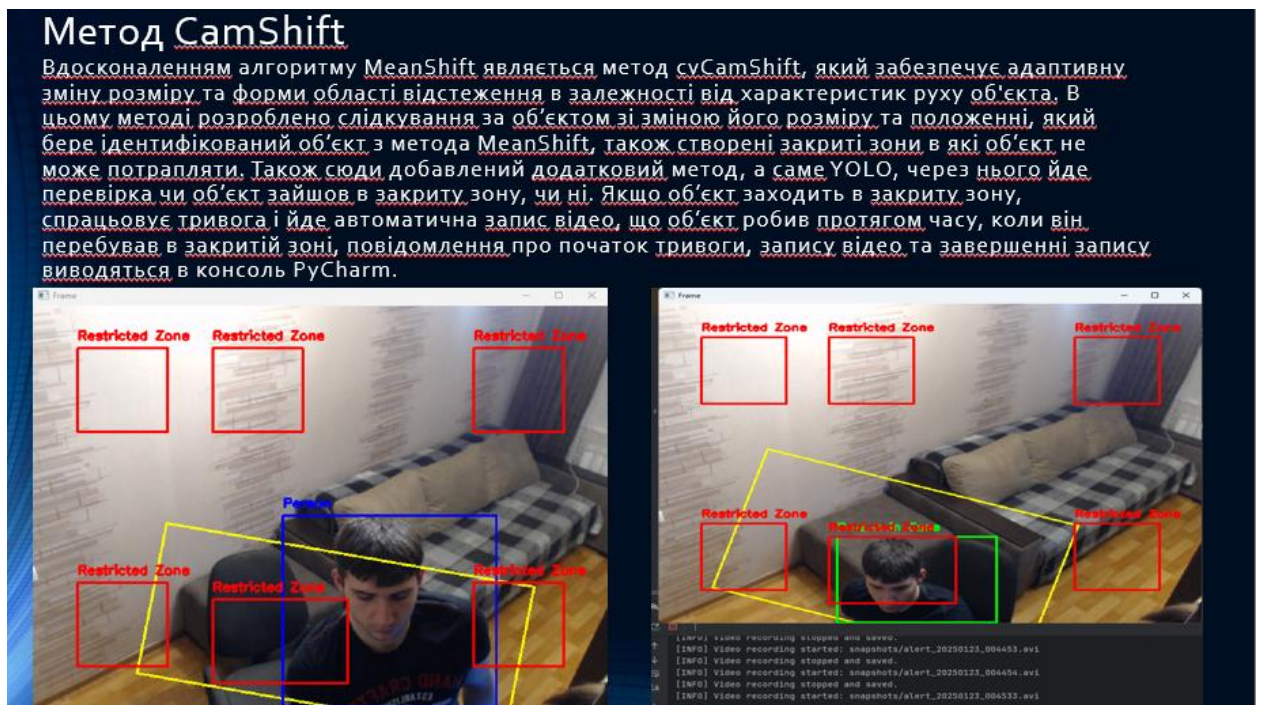


РИСУНОК А.8 – МЕТОД CAMSHIFT

ВИСНОВКИ

Робота спрямована на створення комплексного програмного забезпечення для аналізу руху у відеопотоці, що об'єднує кілька підходів та алгоритмів, кожен із яких реалізовано окремо та адаптовано для певного типу задач. Основними методами є Motion Detection, MeanShift, CamShift та YOLO.

Метод Motion Detection використовується як початковий етап аналізу, дозволяючи фіксувати зони активності у відеопотоці та зберігати фото сумнівних дій об'єкта.

Алгоритм MeanShift забезпечує надійне відстеження об'єктів, аналізуючи їхні кольорові характеристики. Він є ефективним у випадках, коли об'єкт зберігає свої основні візуальні риси, що робить його корисним для задач із невеликою варіативністю об'єктів.

Розширенням цього підходу є CamShift, який адаптує параметри трекінгу залежно від зміни розміру, орієнтації чи форми об'єкта. Це дозволяє використовувати його у складних умовах, де об'єкти можуть змінювати свої характеристики, наприклад, при зміні відстані до камери або повороті.

Окрім класичних методів аналізу, у роботі інтегровано сучасний алгоритм YOLO, який використовується для детекції об'єктів у відео. Завдяки його високій швидкості та точності, програма здатна виявляти кілька об'єктів одночасно, визначаючи їхній тип, координати та розміри. YOLO є ключовим компонентом системи, який забезпечує швидке і ефективне розпізнавання, що особливо важливо для задач у реальному часі.

Об'єднання цих методів у єдиній системі дозволяє створити універсальне рішення для аналізу відео, здатне вирішувати задачі як виявлення, так і трекінгу об'єктів. Користувач має можливість обирати між різними режимами роботи залежно від поставленої задачі, що робить систему гнучкою та придатною для широкого спектра застосувань: від моніторингу безпеки до аналізу руху транспорту чи поведінки людей.

РИСУНОК А.9 – ВИСНОВКИ

