Міністерство освіти і науки України
Національний університет “Львівська політехніка”
Інститут комп’ютерних наук та інформаційних технологій

Кафедра САПР

Лабораторна робота №2

з дисципліни: “Розпізнавання образів і комп’ютерний зір”

на тему:

“Створення маски рухомих об’єктів”

Виконав:
Ст. групи ПП-44
Верещак Б. О.
Прийняв:
Асис. Мельник М. Р.

Львів - 2025

Мета: навчитися виявляти рухомі об’єкти на серії послідовних зображень.

Лабораторне завдання

1. Визначити маску рухомих об’єктів на серії послідовних зображень.
2. Визначити центр маси рухомих об’єктів;

Кроки виконання завдання:

Крок 1: Визначення маски рухомих об’єктів

Першим кроком в даній лабораторній роботі потрібно було визначити маску рухомих об’єктів. Для цього моя програма на початку завантажу є 5 послідовних кадрів, збережених на етапі першої лабораторної роботи.

Для реалізації цього етапу було використано три основні функції:

calcFrameDiff() — обчислює різницю між двома сусідніми кадрами. За допомогою функції absdiff() обчислюється абсолютна різниця пікселів, а потім застосовується порогова фільтрація (threshold()), щоб виділити тільки значні зміни. Таким чином формується двійкова маска руху між двома кадрами.

combineFrameDiffs() — поєднує декілька таких масок, отриманих із різних пар кадрів. Це дозволяє усунути випадкові шумові зміни (наприклад, через освітлення) та отримати більш стабільну загальну маску руху. Перед поєднанням застосовується розмивання (GaussianBlur()) і повторне порогування, щоб зробити результат більш чистим.

cleanMask() — очищує отриману маску від дрібних артефактів та шумів за допомогою морфологічних операцій: MORPH_OPEN — прибирає дрібні точки та шум; MORPH_CLOSE — заповнює дрібні розриви в областях руху, роблячи об’єкти більш суцільними.

На Рис. 1 показано початкове зображення, яке використовується як базовий кадр для аналізу, а на Рис. 2 наведено отриману маску рухомих об’єктів, де білі області відповідають зонам зміни – тобто тим частинам сцени, де зафіксовано рух.

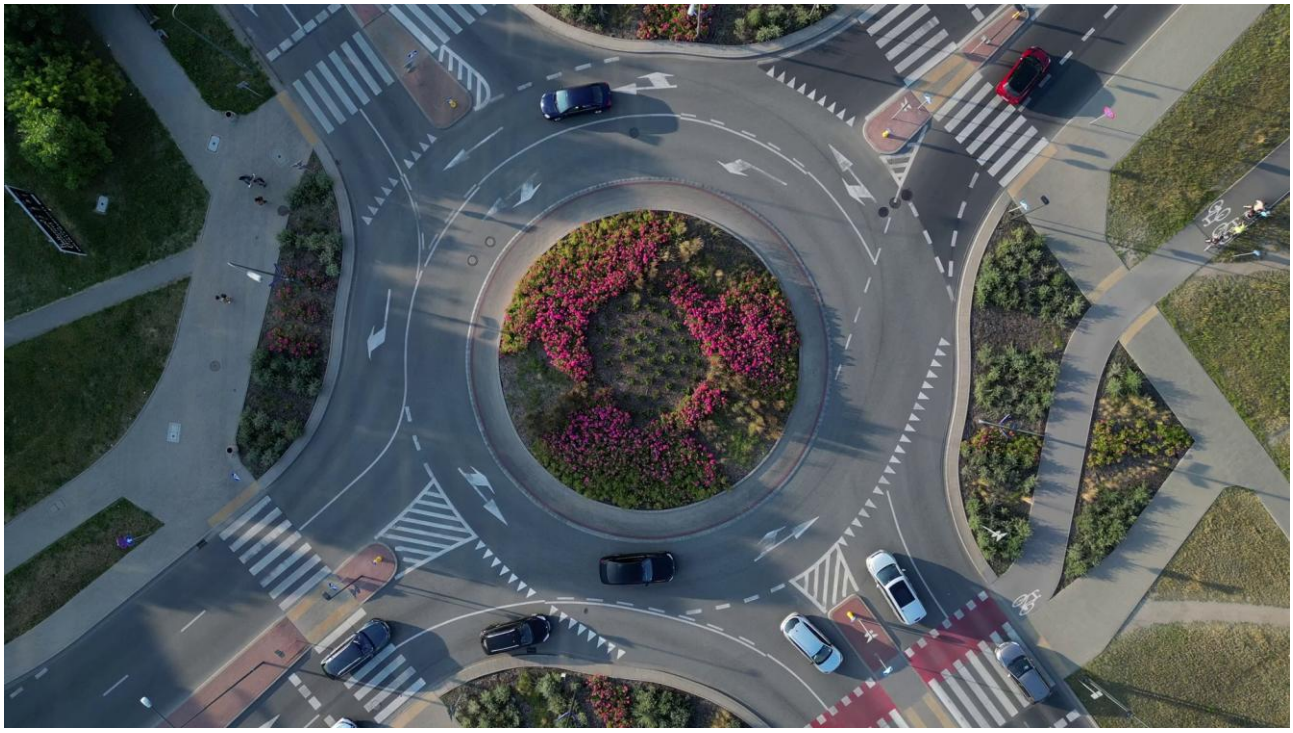

Рис. 1. Початкове зображення з яким будемо порівнювати зміни

Рис. 2. Маска рухомих об’єктів

Код програми:

```
Mat calcFrameDiff(const Mat& frame1, const Mat& frame2, bool useThreshold = true, int threshVal = 10){
    Mat diff;
    absdiff(frame1, frame2, diff);
    if (useThreshold){
        threshold(diff,diff, threshVal, 255, THRESH_BINARY);
    }
```

```
    return diff;
}

Mat combineFrameDiffs(vector<Mat> frames){
    if (frames.empty()) return Mat();

    Mat combined = frames[0].clone();
    for (size_t i = 1; i < frames.size(); ++i) {
        GaussianBlur(combined, combined, Size(3,3),0);
        threshold(combined,combined, 100, 255, THRESH_BINARY);
        bitwise_or(combined, frames[i], combined);
    }

    return combined;
}

Mat cleanMask(Mat mask){
    Mat cleanedMask;
                    morphologyEx(mask,          cleanedMask,          MORPH_OPEN,
getStructuringElement(MORPH_ELLIPSE, Size(5,5)));
                morphologyEx(cleanedMask,       cleanedMask,       MORPH_CLOSE,
getStructuringElement(MORPH_ELLIPSE, Size(5,5)));
    return cleanedMask;
}
```

Крок 2: Визначення центру маси рухомих об’єктів

На цьому етапі основною задачею було визначення центрів маси рухомих об’єктів, отриманих із попереднього кроку. Для цього я використав функції, які дозволяють знайти контури об’єктів та обчислити їхні моменти, з яких визначаються координати центрів маси. Я вирішив додатково побудувати обмежувальні прямокутники навколо кожного виявленого об’єкта.

У чистій масці руху відбувається пошук контурів, які відповідають межам рухомих об’єктів. Для цього використовується функція findContours з параметром RETR_EXTERNAL, що дозволяє отримати лише зовнішні контури без вкладених, та CHAIN_APPROX_SIMPLE, який апроксимує контур лінійними сегментами і зменшує кількість точок.

Для кожного виявленого контуру обчислюються його характеристики. Площа контуру визначається за допомогою contourArea, і якщо вона менша за встановлене мінімальне значення minArea, контур вважається шумом і відкидається. Моменти контуру обчислюються функцією moments, що дає змогу

визначити нульовий момент m00, пропорційний площі контуру, та перші моменти m10 і m01. Центр маси об’єкта визначається як співвідношення centerX = m10/m00 та centerY = m01/m00. Для наочності рухомих об’єктів обчислюється обмежувальний прямокутник за допомогою boundingRect, який визначає найменший прямокутник, що повністю містить контур.

Було реалізована функція, яка приймає початковий кадр, контури об’єктів і обчислені центри маси, і повертає зображення з накладеними зеленими прямокутниками та синіми точками центрів. У результаті отримуємо кадр, на якому можна легко бачити не лише місцезнаходження рухомих об’єктів, а й їхні межі та центри маси, що значно полегшує подальший аналіз руху.

На Рис. 3 показано результат цього кроку: сині точки позначають центри маси, а зелені прямокутники окреслюють межі кожного об’єкта, що рухається.

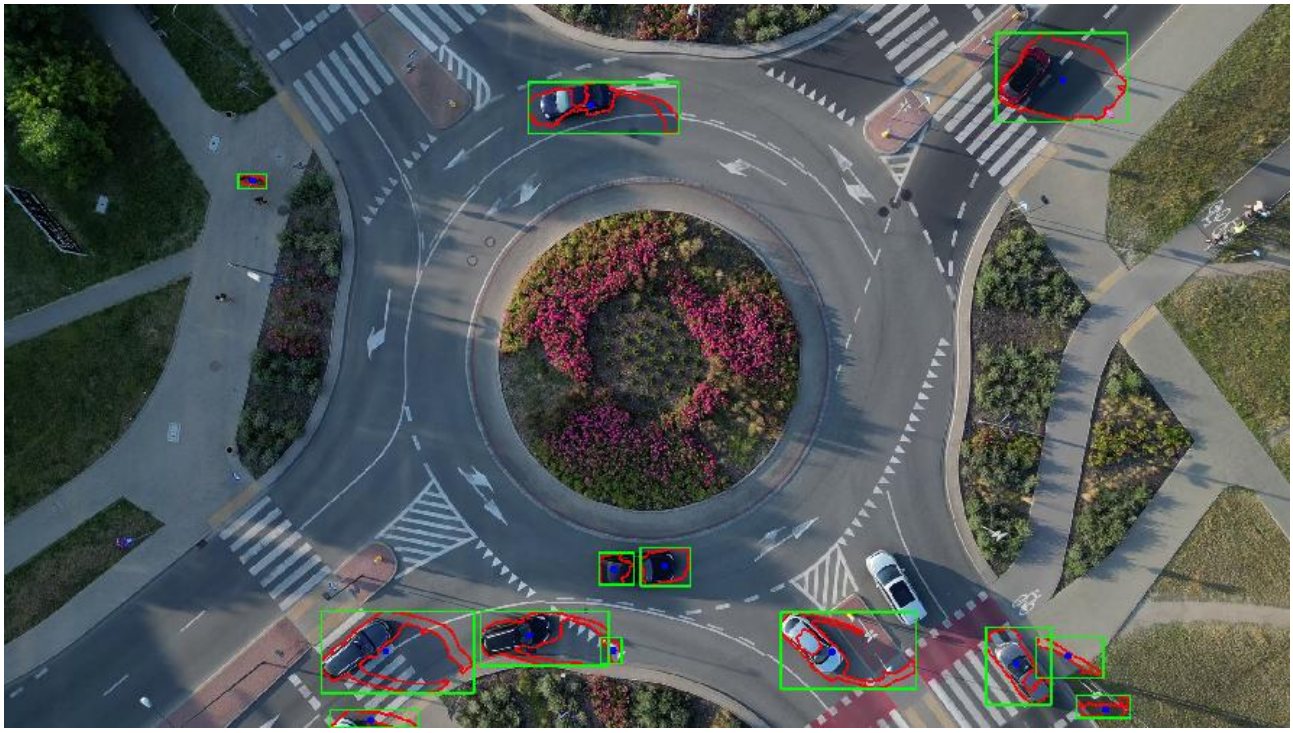

Рис. 3. Обмежувальні прямокутники з центром маси рухомих об’єктів

Код програми:

```
vector<vector<Point>> findMovingObjContours(Mat frame, double minArea = 500.0){
   vector<vector<Point>> contours;
   findContours(frame, contours, RETR_EXTERNAL, CHAIN_APPROX_SIMPLE);
   vector<vector<Point>> filtered;
   for (const auto& c : contours) {
      if (contourArea(c) > minArea) {
         filtered.push_back(c);
```

```
        }
    }
    return filtered;
}

vector<Point2f> getContourCenters(const vector<vector<Point>>& contours) {
    vector<Point2f> centers;
    for (const auto& c : contours) {
        Moments m = moments(c);
        if (m.m00 != 0) {
            centers.push_back(Point2f(float(m.m10/m.m00), float(m.m01/m.m00)));
        }
    }
    return centers;
}

Mat drawContoursAndCenters(const Mat& frame, const vector<vector<Point>>& contours, const vector<Point2f>& centers, bool isGray = true) {
    Mat output;
    if (isGray)
        cvtColor(frame, output, COLOR_GRAY2BGR);
    else
        output = frame.clone();

    drawContours(output, contours, -1, Scalar(0,0,255), 2);
    for (const auto& c : contours) {
        Rect bbox = boundingRect(c);
        rectangle(output, bbox, Scalar(0,255,0), 2);
    }
    for (const auto& center : centers) {
        circle(output, center, 5, Scalar(255,0,0), -1);
    }
    return output;
}

int main() {
    VideoCapture cap("Traffic.mp4");
    if (!cap.isOpened()) {
        cout << "Cant open video!" << endl;
        return -1;
    }

    namedWindow("Video", WINDOW_NORMAL);
    resizeWindow("Video", 800, 600);

    Mat frame;
    while (true) {
        cap >> frame;
```

```
        if (frame.empty()) break;

        imshow("Video", frame);
        if (waitKey(30) == 27)
            break;
    }

    cap.set(CAP_PROP_POS_FRAMES, 0);

    vector<string> filenames = saveFrames(cap, 10);
    vector<Mat> frames;
    vector<Mat> frameDiffs;

    for (int i = 0; i < 5; i++){
        frames.push_back(openImage(filenames[i], false, true));
    }

    for (int i = 1; i < frames.size(); ++i) {
        frameDiffs.push_back(calcFrameDiff(frames[i-1], frames[i], true));
    }

    Mat motionMask = combineFrameDiffs(frameDiffs);
    motionMask = cleanMask(motionMask);
    namedWindow("Combined Diff", WINDOW_NORMAL);
    imshow("Combined Diff", motionMask);
    waitKey(0);

    vector<vector<Point>> contours = findMovingObjContours(motionMask);
    vector<Point2f> centers = getContourCenters(contours);
    Mat background = openImage(filenames[0], false, false);
    Mat result = drawContoursAndCenters(background, contours, centers, false);
    namedWindow("Motion Detection", WINDOW_NORMAL);
    imshow("Motion Detection", result);
    waitKey(0);

    destroyAllWindows();
    return 0;
}
```

Висновки

Лабораторна робота успішно продемонструвала методи детектування рухомих об’єктів на основі порівняння послідовних кадрів відео. Реалізований алгоритм, побудований на принципі розрахунку різниці кадрів, комбінування множинних різниць та морфологічної обробки, показав ефективність у виявленні динамічних об’єктів на відеозаписі трафіку.

Через відсутність умов добре освіченої сцени, так як всі об’єкти на відео мали свої тіні, я не зміг досягти ідеального результату розпізнавання об’єктів. Виявлені об’єкти мають не дуже коректні центри мас та обмежувальні прямокутники, але все одно дозволяють оцінити ефективність використаних методів.

Отримані навички роботи з морфологічною обробкою зображень, пошуком контурів та розрахунком моментів об’єктів є фундаментальними для розробки більш складних систем комп’ютерного зору. Розуміння взаємодії між чутливістю виявлення та специфічністю (мінімізацією помилкових спрацювань) є важливим для оптимізації алгоритмів детектування в різних сценаріях застосування.