Захист <https://youtu.be/b4-I6XacKRs>

**Лабораторна робота №3**

**ІДЕНТИФІКАЦІЯ ОБ’ЄКТІВ НА ЗОБРАЖЕННІ**

**Мета:** навчитися розробляти і реалізовувати алгоритми ідентифікації об'єктів графічних зображень.

**3.1 Теоретичні відомості**

Під ідентифікацією об'єктів зображення розуміють, як визначення їх геометричних властивостей, так і віднесення об'єктів до якої-небудь, заздалегідь визначеної групи об'єктів.

У цій лабораторній роботі, необхідно навчитися виробляти виділення первинних ознак об'єктів графічного зображення з метою подальшої класифікації об'єктів в рамках певної предметної області.

Нижче розглянемо кілька прийомів, які дозволяють виміряти значення ознак об'єктів зображення. Алгоритми наводяться лише для ознайомлення.

Часто треба розрізнити об’єкти на зображенні по кількості отворів, або інородних включень. для цього можна застосувати наступній алгоритм

Крок 1: Зчитування зображення.

Крок 2: Для кожного об‘єкту визначити його межі ( координати мінімалього прямокутника (Р), який він займає) та скопіювати область Р для подальших дій.

Крок 3: Видалити з області Р усі інші об‘єкти окрім досліджуваного

Крок 4: Підрахувати число Ейлера для очищеної області Р.

Крок 5: В залежності від значення числа Ейлера прийняти рішення щодо позначення об’єкта символом.

Інколи треба розрізнити об’єкти на зображенні по кількості кутів. для цього можна застосувати наступній алгоритм

Крок 1: Зчитування зображення.

Крок 2: Для кожного об‘єкту визначити його граничні пікселі .

Крок 3:. Серед граничних пікселів знай такі, число сусідів, яких буде менше половини (<4)/ Такі пікселі й визначають кількість гострих кутів у об‘єкта.

Крок 4: Підрахувати кількість N кутових пікселів.

Крок 5:В залежності від N прийняти рішення щодо позначення об’єкта символом.

Інші варіанти завдань в даній лабораторній роботі потребують або комбінованого застосування наданих алгоритмів, або розробки інших евристичних алгоритмів.

**Хід роботи**

**14 Реалізувати програму, що відрізняє прямокутники та трикутники від їнших об‘єктів, також позначає їх сімволами «+» та «-« та підраховує іх кількість на зображенні. Тестове зображення створити самостійно.**

В лістингу 1 зображено код програми.

Лістинг 1

import cv2

number = 0

img = cv2.imread('1.png')

# перетворити зображення в градації сірого

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

# застосувати порогове значення для перетворення зображення у градаціях сірого на двійкове зображення

ret, thresh = cv2.threshold(gray, 50, 255, 0)

# знайти контури

contours, hierarchy = cv2.findContours(

    thresh, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)

for cnt in contours:

    approx = cv2.approxPolyDP(cnt, 0.01\*cv2.arcLength(cnt, True), True)

    if len(approx) == 3:

        number += 1

        img = cv2.drawContours(img, [cnt], -1, (255, 255, 0), 3)

        # обчислення центр мас трикутника

        M = cv2.moments(cnt)

        if M['m00'] != 0.0:

            x = int(M['m10']/M['m00'])

            y = int(M['m01']/M['m00'])

        cv2.putText(img, '-', (x, y), cv2.FONT\_HERSHEY\_SIMPLEX,

                    0.6, (255, 255, 0), 2)

    elif len(approx) == 4:

        number += 1

        img = cv2.drawContours(img, [cnt], -1, (255, 255, 0), 3)

        # обчислення центр мас прямокутника

        M = cv2.moments(cnt)

        if M['m00'] != 0.0:

            x = int(M['m10']/M['m00'])

            y = int(M['m01']/M['m00'])

        cv2.putText(img, '+', (x, y),

                    cv2.FONT\_HERSHEY\_SIMPLEX, 0.6, (255, 255, 255), 2)

print("Знайдено потрібних нам об'єктів:", number)

cv2.imshow("Shapes", img)

cv2.waitKey(0)

cv2.destroyAllWindows()

На рисунку 1 зображено результат виконання програми, а саме обведення трикутників і прямокутників на картинці, та підрахунок їх кількості.

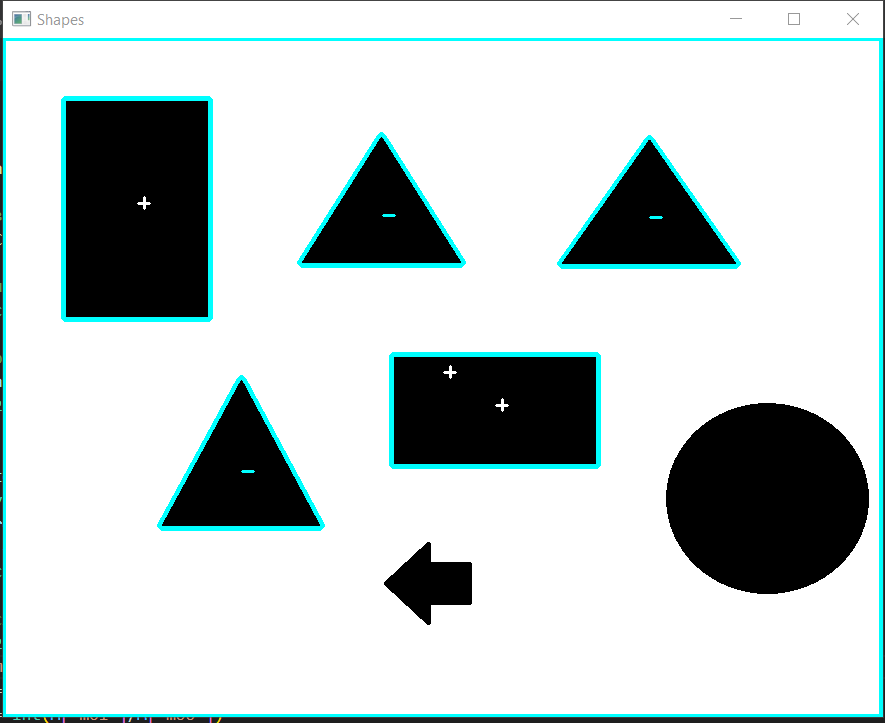


Рисунок 1 – Результат роботи коду

**Висновок:** розроблено програму із застосуванням стандартних функцій середовища Matlab, для реалізації функції вибору об'єктів.