Захист [https://youtu.be/IryLh5ZKygg](%20https://youtu.be/IryLh5ZKygg)

**Лабораторна робота №2**

**РЕАЛІЗАЦІЯ ФУНКЦІЙ ОБРОБКИ ГРАФІЧНИХ ЗОБРАЖЕНЬ**

**Мета:** самостійно розробити алгоритм і провести його реалізацію, із застосуванням стандартних функцій середовища Matlab, без використання спеціалізованих методів пакету Image Processing Toolbox.

**2.1 Теоретичні відомості**

**2.1.1 Складні об”єкти**

Інколи в поле зору потрапляє більше одного об'єкта (рис. 2.1). У цьому випадку обчислення площі, геометричного центру та орієнтації призведе до значень, "усереднених" за всіма компонентами бінарного зображення. Як правило, це не те, що потрібно. Бажано якось помітити окремі компоненти зображення і обчислити значення площі, перших і других моментів для кожної компоненти окремо.

**2.1.2 Розмітка компонент**

Будемо вважати дві точки зображення пов'язаними, якщо існує шлях між ними, вздовж якого характеристична функція постійна

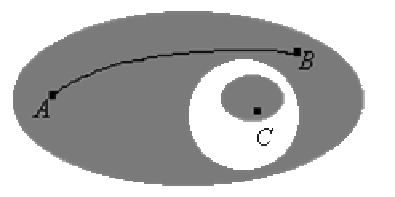


Рисунок 2.1 – Зображення, що складається з декількох областей, для кожної з яких необхідно проводити розрахунок положення і орієнтації .

Так, на рисунку 2.1 точка А пов'язана з точкою В, але не пов'язана з точкою С. Зв'язана компонента бінарного зображення є максимальною множиною пов'язаних точок, тобто множина, що складається з усіх тих точок, між двома з яких існує зв'язуючий їх шлях.

Елементи зображення необхідно помітити таким чином, щоб ті з них, які належать одній області були відмінні від інших. Для цього нам необхідно вирішити, які точки належать одній і тій же області. На рисунку 2.1 точка А вважається пов'язаною з точкою В, оскільки ми можемо знайти безперервну криву, яка цілком лежить в затіненій області та сполучає зазначені точки. Ясно, що точка А не пов'язана з точкою С, так як в цьому випадку не можливо знайти подібної кривої.

Один із способів розмітки об'єктів на дискретному бінарному зображенні полягає у виборі довільної точки, в якій bij = 1. Цій точці та її сусідам приписують певну мітку. На наступному кроці позначаються сусіди цих сусідів (крім уже помічених) і т.д. По завершенні цієї рекурсивної процедури, одна компонента буде повністю позначена, і процес можна буде продовжити, вибравши нову початкову точку та нову мітку для позначки. Щоб її відшукати, досить яким-небудь систематичним чином переміщуватися по зображенню до тих пір, поки не зустрінеться остання непомічена точка, в якій bij = 1. Коли на цьому етапі не залишиться жодного такого елемента, всі об'єкти зображення виявляться розміченими.

Ясно, що "фон" також можна розбити на зв'язкові компоненти, оскільки об'єкти можуть мати отвори. Їх можна помітити за допомогою тієї ж процедури, але при цьому необхідно звертати увагу не на одиниці, а на нулі.

**2.1.3 Зв'язаність**

Тепер потрібно розглянути зміст терміну сусід. Якщо ми маємо справу з квадратним растром, то, сусідами слід вважати чотири елементи зображення, що торкаються сторін даного елемента. Але як бути з тими, що торкаються його в кутах? Існують дві можливості:

• чотирьохзв'язність - сусідами вважаються тільки елементи, які примикають до сторін;

• восьмизв'язність - елементи, які торкаються в кутах, також вважаються сусідами. Зазначені можливості наведені на рисунку 2.2:

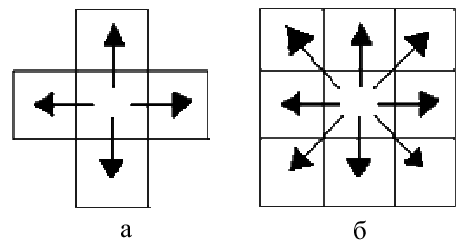


Рисунок 2.2 – Зв'язність елементів: а) чотирьохзв'язність, б) восьми зв'язність.

Виявляється, жодна з цих схем не є повністю задовільною. У цьому можна переконатися, якщо згадати, що фон також можна розбити на кілька зв'язаних компонент. Тут нам хотілося б застосувати наші інтуїтивні уявлення про зв'язність областей на безперервному бінарному зображенні. Так, наприклад, проста замкнута крива повинна розділяти зображення на дві зв'язані області (рис. 2.3). Це так звана теорема Жордана про криву.

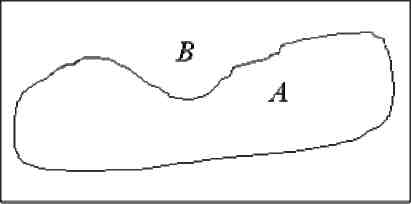


Рисунок 2.3 Проста замкнута крива, розбиває площину на дві зв'язні області.

Тепер розглянемо просте зображення, що містить чотири елементи зі значенням "одиниця", які примикають до центрального елементу зі значенням "нуль":

0 1 0

1 0 1

0 1 0

Це - хрест з викинутим центром. Якщо ухвалити угоду про чотирьохзв'язності, то на зображенні виявляться чотири різні компоненти (O1, O2, O3 і O4):

B1 O1 B1

O2 B2 O3

B1 O4 B1

Вони, природно, не утворюють замкнутої кривої, хоча центральний елемент, що відноситься до фону, і не пов'язаний з рештою фону. Незважаючи на відсутність будь-якої замкнутої кривої, у нас утворилися дві фонові області. Якщо ухвалити угоду про восьмизв‘язність, то, навпаки, чотири елементи растра будуть утворювати замкнуту криву, і в той же час центральний елемент виявиться пов'язаним з рештою фоном:

B O B

O B O

B O B

Отже, ми отримали замкнуту криву і тільки одну зв'язану компоненту фону.

Одне з рішень виниклої проблеми полягає у виборі чотирьохзв'язності для об'єкта і восьмизв'язності для фону (або навпаки). Така асиметрія в трактуванні об'єкта і фону часто небажана, і її можна уникнути шляхом введення іншого типу асиметрії. Будемо вважати сусідами чотири елементи зображення, що примикають до даного по сторонах, а також два з чотирьох елементів, що торкаються в кутах:



Рисунок 2.4 –Шестизв'язного елементів

Для забезпечення симетричності відносини зв'язності два кутові елемента повинні перебувати на одній і тій же діагоналі: якщо елемент А - сусід елемента В, то елемент В повинен бути сусідом елемента А. Найчастіше користуються першим з двох можливих варіантів, наведених вище. За допомогою шестизв'язності як об'єкт, так і фон можна трактувати одноманітно без будь-яких подальших непогодженостей. Така зв'язність використовується для зображень на квадратному растрі.

**2.2 Локальні обчислення**

Дотепер основна увага приділялася послідовній обробці інформації, що міститься в бінарному зображенні. Тепер ми розглянемо деякі алгоритми, які можуть використовувати паралельні обчислення.

**2.2.1 Число Ейлера**

Число Ейлера визначається як різниця між кількістю об'єктів на зображенні та кількістю дірок в цих об'єктах. Параметр n визначає використовуваний критерій зв'язності. Він може бути рівним 4 або 8. При виконанні функції bweuler параметр n можна опустити, в цьому випадку об'єкти розглядаються як 8-зв'язкові.

*Алгоритм:*

Ідея алгоритму полягає у підрахунку кількості квадратних матриць 2х2 з різною кількостю ненульових елементів. Потім за допомогою формул, отриманих емпіричним шляхом підрахуємо число Ейлера.

Для спрощення підрахунків будемо використовувати наступний підхід. Введемо наступні позначення:













Тоді для 4-зв'язаних об'єктів число Ейлера визначається за формулою з [1]:

 (2.1)

а для 8-зв'язкових - за формулою

 (2.2)

**2.2.2 Обчислення периметру**

Обчислення периметру являє собою лише приблизну оцінку, оскільки, як правило, дискретне бінарне зображення будується на основі безперервного, і при цьому межі об'єктів стають більш порізаними. Наприклад, оцінка довжини діагональної прямої в 2 разів більше "істинної":

Усереднення по всіх кутках нахилу дає середнє значення коефіцієнта, що показує, у скільки разів збільшено отримане значення. Воно складає 4 / π = 1,273 .... Розділивши на це число, можна поліпшити оцінку периметра.

**2.2.3 Побудова остова (скелета) об'єкта**

Операція побудова остова (скелета) об'єкта виконуєця із застосовуанням ерозії об'єкта з урахуванням ряду умов для збереження 8-зв'язності остова. У результаті послідовного застосування даної операції можна побудувати остов, що представляє собою зв'язну лінію товщиною в 1 піксель, що проходить по середині об'єкта:/

Бінарні зображення можна комбінувати різними шляхами. Можна здійснити операцію АБО. У результаті ми об'єднаємо два зображення в одне. Великий інтерес представляє те, як характеристики одержуваних подібними способами зображень співвідносяться з характеристиками вихідних зображень. Одна з причин такого інтересу пов'язана з надією розбити зображення на велику кількість частин, одночасно обробити всі ці частини і потім об'єднати результат.

**Хід роботи**

14 Видалити всі об'єкти, площа яких менше заданої кількості пікселів.

Створено програму на мові програмування Python. Програма видаляє всі об'єкти, площа яких менше заданої кількості пікселів. В лістингу 1 зображено код програми.

Лістинг 1

import cv2

# Завантажити зображення, перетворити на градації сірого, розмиття за Гауссом, поріг Оцу

image = cv2.imread('1.jpg')

#image = cv2.imread('2.png')

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

blur = cv2.GaussianBlur(gray, (3, 3), 0)

thresh = cv2.threshold(

    blur, 0, 255, cv2.THRESH\_BINARY\_INV + cv2.THRESH\_OTSU)[1]

# Фільтруйте за допомогою контурної області та видаляйте невеликий шум

cnts = cv2.findContours(thresh, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)

cnts = cnts[0] if len(cnts) == 2 else cnts[1]

for c in cnts:

    area = cv2.contourArea(c)

    if area < 20000:  # площа

        cv2.drawContours(thresh, [c], -1, (0, 0, 0), -1)

# Морфінг закрити та інвертувати зображення

kernel = cv2.getStructuringElement(cv2.MORPH\_RECT, (5, 5))

close = 255 - cv2.morphologyEx(thresh, cv2.MORPH\_CLOSE, kernel, iterations=2)

cv2.imshow('thresh', thresh)

cv2.imshow('close', close)

cv2.waitKey()

На рисунку 1 і 2 зображено вхідну картинку. На рисунку 3 і 4 результат обробки програмою картинок і видалення елементів які менші в площі за задану кількість пікселів.



Рисунок 1 – Вхідна картинка

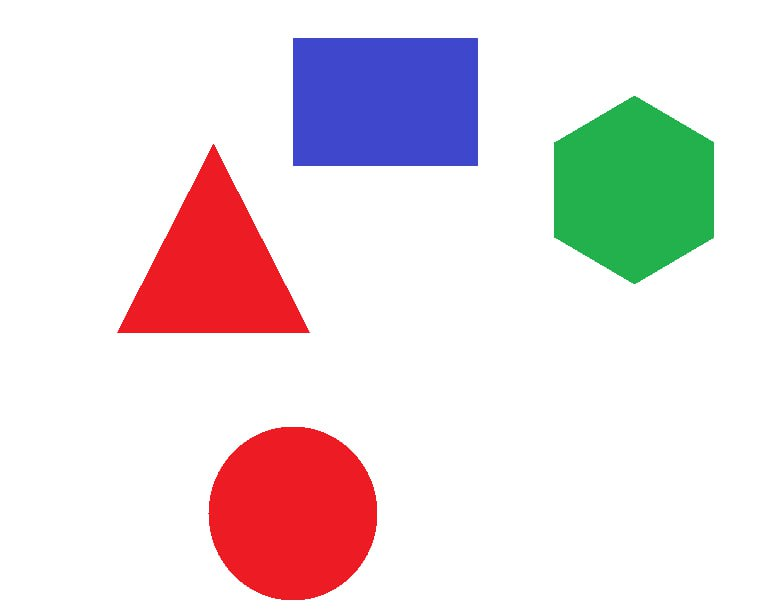


Рисунок 2 – Вхідна картинка

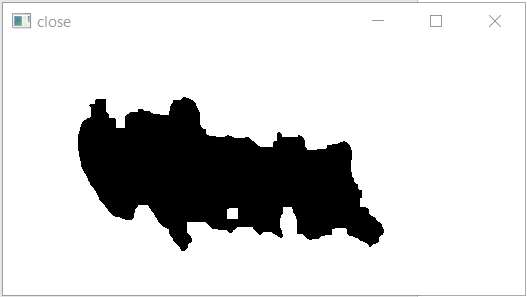


Рисунок 3 – Результат видалення

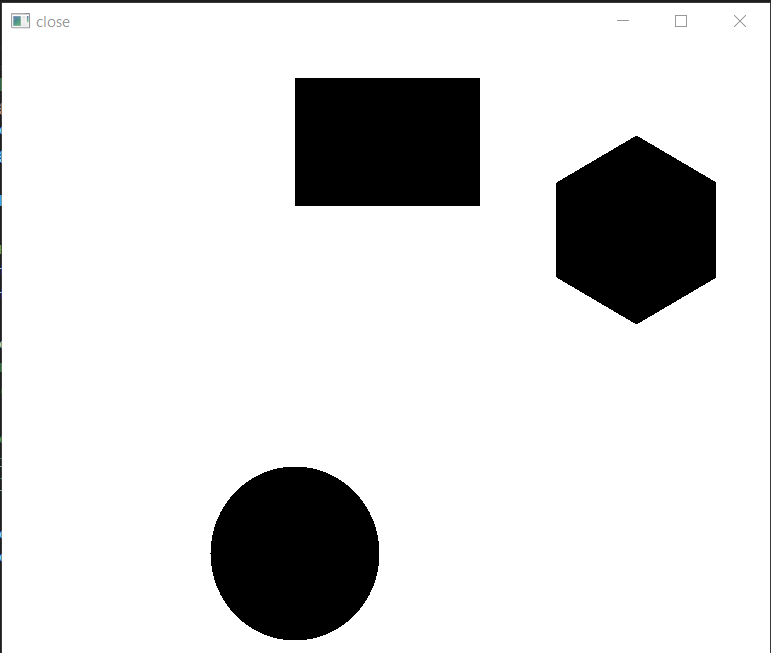


Рисунок 4 – Результат видалення

**Висновок:** Було розроблено програму із застосуванням стандартних функцій середовища Matlab, для реалізації функції видалення обєктів площа яких менше программно заданої кількості пікселів.