**Міністерство освіти і науки України**

**Національний технічний університет України «КПІ» імені Ігоря Сікорського**

**Кафедра обчислювальної техніки ФІОТ**

**ЗВІТ**

**з лабораторної роботи № 5**

**з навчальної дисципліни «Computer Vision»**

**Тема:**

**ДОСЛІДЖЕННЯ ТЕХНОЛОГІЙ ПОКРАЩЕННЯ ЯКОСТІ ЦИФРОВИХ ЗОБРАЖЕНЬ ДЛЯ ЗАДАЧ COMPUTER VISION**

**Виконав:**

Студент 3 курсу кафедри ІПІ ФІОТ,

Навчальної групи ІП-14

Бабіч Д. В.

**Перевірив:**

Професор кафедри ОТ ФІОТ

Писарчук О. О.

**Київ 2024**

1. **Мета:**

Дослідити принципи та особливості практичного застосування технологій сегментації та кластеризації цифрових зображень для задач Computer Vision з використанням спеціалізованих програмних бібліотек.

1. **Завдання:**

**Завдання ІІ рівня.**

Функціонал скрипта реалізовано у повному обсязі, п. 6 технічних вимог (ідентифікація об’єкта) реалізовано шляхом програмного порівняння контурів.

Таблиця 1.1 – Варіант завдання

| **Варіант**  (місяць народження) | **Контент цифрового зображення / джерело** | **Технічні умови** |
| --- | --- | --- |
| 7-12 | Оперативні: <https://livingatlas2.arcgis.com/landsatexplorer/>  Високоточні: https://www.bing.com/maps | Район спостереження – обрати самостійно. Об’єкти ідентифікації – обрати самостійно. Дата оперативних даних – обрати самостійно. Метод і технологія сегментації / кластеризації – повинні забезпечувати можливість розрізнення та ідентифікацію обраних об’єктів спостереження. |

1. **Результати виконання лабораторної роботи.**
2. **Синтезована математична модель обробки графічних зображень.**

На стадії попередньої обробки зображення використовуються морфологічні операції, такі як закриття, ерозія та дилатація, для видалення непотрібних дрібних деталей у зображенні та з’єднання близьких областей відокремлених частин. Логічні формули даних морфологічних операцій:

Ерозія: ; Дилатація:

Визначення ерозії, де – вхідне зображення, – структурний елемент, заданий ядром.

Під час основної стадії процесу кластеризації створюється маска, що визначає області, які потрібно розглядати у подальшому аналізі завдяки обранню тих частин зображення, які перебувать у заданих межах по кольору.

Де – координати накладено маски – мінімальна межа діапазону кольору, – максимальна межа діапазону кольору.

Потім знаходяться контури об’єктів на зображенні. Для кожного контуру визначається прямокутник, що охоплює його. Якщо розміри та співвідношення сторін прямокутника відповідають заданим параметрам, прямокутник додається до списку. Для кожного прямокутника на зображенні накладається прямокутник з заданим кольором та товщиною.

1. **Результати архітектурного проектування та їх опис.**



Рисунок 1.1 – Діаграма алгоритму ідентифікації

1. Використання морфологічних операцій з метою видалення дрібних деталей та з’єднання контурів;
2. Застосування маски, яка виконує кластеризацію зображення на основі меж кольору;
3. Виявлення контурів об’єктів;
4. Визначення об’єктів та перевірка чи підходять вони під задані величини;
5. Відображення контурів на зображенні.
6. **Опис структури проекту програми.**

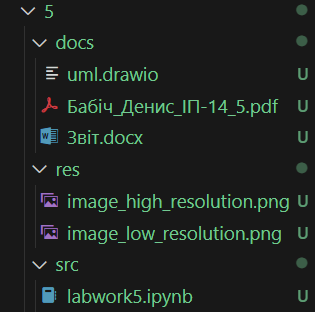


Рисунок 1.2 – Структура проєкту

У директорії src зберігається labwork5.py, який є юпітер-записником з вихідним кодом, директорія docs – зберігає файли звіту у форматі pdf, docx та drawio (файл з діаграмами), директорія res – 2 зображення того ж місця у високій якості та низькій.

1. **Результати роботи програми відповідно до завдання.**



Рисунок 1.3 – Початкове зображення у високій якості

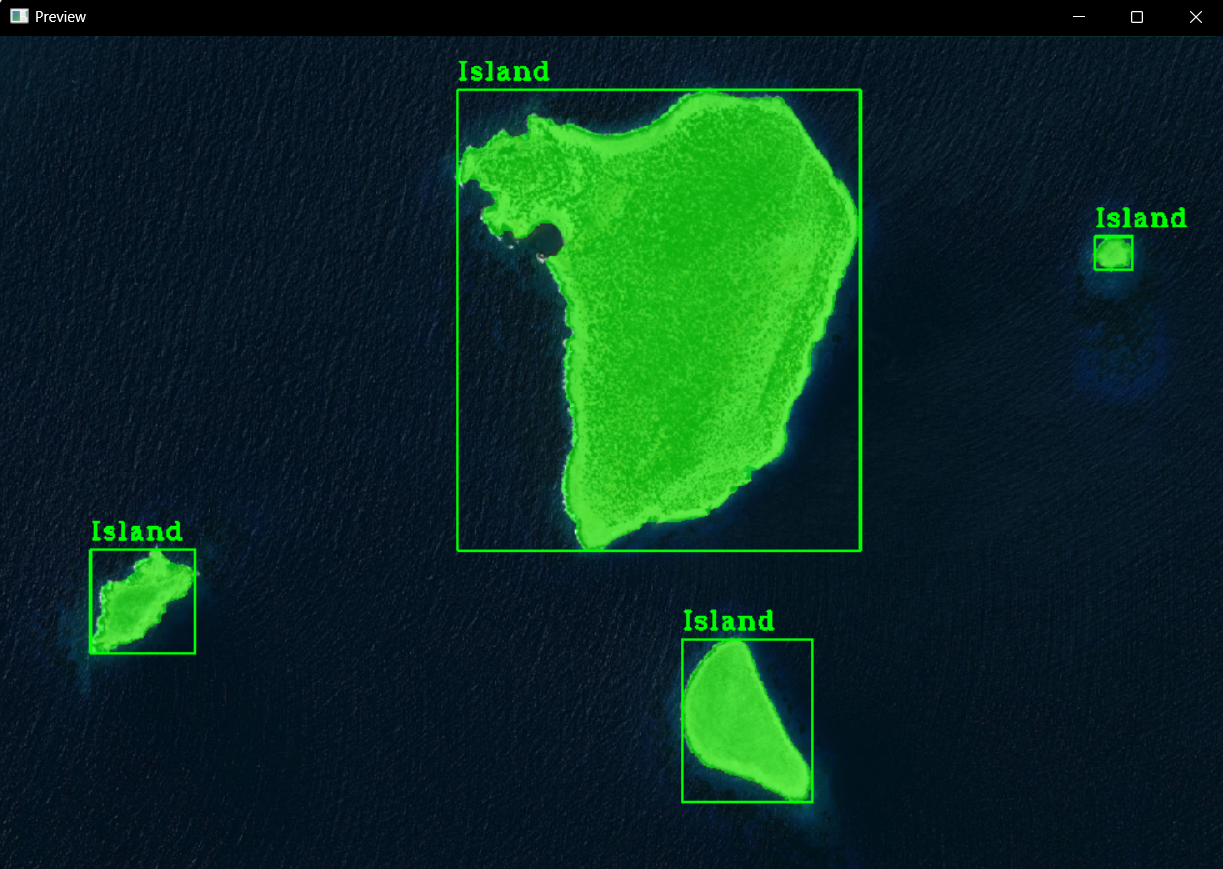


Рисунок 1.4 – Результат виконання кластеризації



Рисунок 1.5 – Початкове зображення низької якості

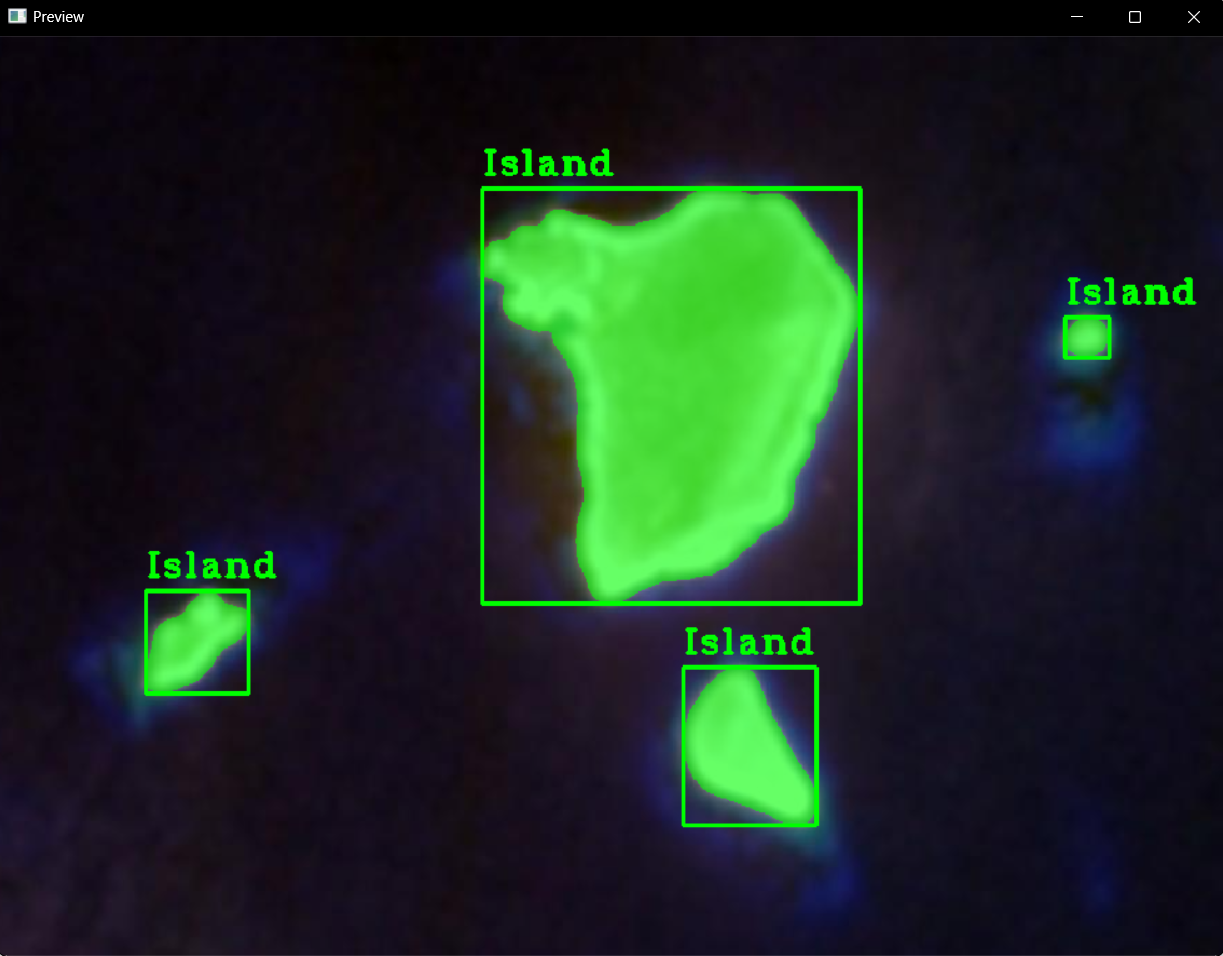


Рисунок 1.6 – Результат кластеризації

1. **Програмний код, що забезпечує отримання результату.**

# %% [markdown]

# # Лабораторна робота № 5

# %% [markdown]

# ## ІП-14 Бабіч Денис (09.07.2003)

# %% [markdown]

# ---

# %% [markdown]

# # Підготовчий етап

# %% [markdown]

# ## Імпортування необхідних модулів

# %%

import cv2

import numpy as np

# %% [markdown]

# ## Створення допоміжних функцій

# %%

def load\_image(relative\_path: str) -> np.array:

return cv2.imread(relative\_path)

def show\_image(image: np.array, title: str = "Preview") -> None:

cv2.namedWindow(title, cv2.WINDOW\_NORMAL)

cv2.imshow(title, image)

cv2.waitKey(0)

cv2.destroyAllWindows()

def preprocess\_image(image: np.ndarray, kernel\_size: int) -> np.ndarray:

kernel = cv2.getStructuringElement(cv2.MORPH\_ELLIPSE, (kernel\_size, kernel\_size))

preprocessed\_frame = cv2.morphologyEx(image, cv2.MORPH\_CLOSE, kernel)

preprocessed\_frame = cv2.erode(preprocessed\_frame, kernel, iterations = 1)

preprocessed\_frame = cv2.dilate(preprocessed\_frame, kernel, iterations = 1)

return preprocessed\_frame

def apply\_color\_filtering\_mask(image\_stock: np.array, image\_preprocessed: np.array, exclude\_color\_bottom\_bound: tuple, exclude\_color\_upper\_bound: tuple, roi\_color: tuple = (0, 255, 0), roi\_transparency: int = 60) -> tuple:

mask = cv2.bitwise\_not(cv2.inRange(image\_preprocessed, exclude\_color\_bottom\_bound, exclude\_color\_upper\_bound))

contours, \_ = cv2.findContours(mask, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)

overlay = image\_stock.copy()

cv2.fillPoly(overlay, contours, roi\_color)

cv2.addWeighted(overlay, roi\_transparency / 100, image\_stock, 1 - roi\_transparency / 100, 0, image\_stock)

return (image\_stock, mask)

def overlay\_objects\_contours(image\_stock: np.array, mask: np.array, width\_min: int, height\_min: int, width\_max: int, height\_max: int, roi\_title: str, roi\_color: tuple = (0, 255, 0), roi\_thickness: int = 2) -> np.ndarray:

ROI\_TITLE\_OFFSET = 10

FONT\_SCALE = 1.25

contours, \_ = cv2.findContours(mask, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)

for contour in contours:

x, y, width, height = cv2.boundingRect(contour)

if (width\_min < width < width\_max) and (height\_min < height < height\_max):

cv2.rectangle(image\_stock, (x, y), (x + width, y + height), roi\_color, roi\_thickness)

cv2.putText(image\_stock, roi\_title, (x, y - ROI\_TITLE\_OFFSET), cv2.FONT\_HERSHEY\_COMPLEX\_SMALL, FONT\_SCALE, roi\_color, roi\_thickness)

return image\_stock

# %% [markdown]

# ---

# %% [markdown]

# # Основний етап

# %% [markdown]

# ## Зображення з високою роздільною здатністю

# %%

KERNEL\_SIZE = 10

COLOR\_BOTTOM\_BOUND = (0, 0, 0)

COLOR\_UPPER\_BOUND = (255, 255, 50)

WIDTH\_MIN = 10

HEIGHT\_MIN = 10

WIDTH\_MAX = 1000

HEIGHT\_MAX = 1000

ROI\_TITLE = "Island"

image = load\_image("../res/image\_high\_resolution.png")

preprocessed\_image = preprocess\_image(image, KERNEL\_SIZE)

processed\_image, mask = apply\_color\_filtering\_mask(image, preprocessed\_image, COLOR\_BOTTOM\_BOUND, COLOR\_UPPER\_BOUND)

processed\_image = overlay\_objects\_contours(image, mask, WIDTH\_MIN, HEIGHT\_MIN, WIDTH\_MAX, HEIGHT\_MAX, ROI\_TITLE)

show\_image(processed\_image)

# %% [markdown]

# ## Зображення з низькою роздільною здатністю

# %%

KERNEL\_SIZE = 5

COLOR\_BOTTOM\_BOUND = (0, 0, 0)

COLOR\_UPPER\_BOUND = (255, 255, 100)

WIDTH\_MIN = 10

HEIGHT\_MIN = 10

WIDTH\_MAX = 1000

HEIGHT\_MAX = 1000

ROI\_TITLE = "Island"

image = load\_image("../res/image\_low\_resolution.png")

preprocessed\_image = preprocess\_image(image, KERNEL\_SIZE)

processed\_image, mask = apply\_color\_filtering\_mask(image, preprocessed\_image, COLOR\_BOTTOM\_BOUND, COLOR\_UPPER\_BOUND)

processed\_image = overlay\_objects\_contours(image, mask, WIDTH\_MIN, HEIGHT\_MIN, WIDTH\_MAX, HEIGHT\_MAX, ROI\_TITLE)

show\_image(processed\_image)

1. **Висновки.**

У цій роботі було проведено дослідження різних підходів для роботи із зображеннями високої та роздільної здатності з метою виконання колірної кластеризації та пошуку об’єктів на них. Для виконання завдання були залучені такі методи роботи з растровими зображеннями, як морфологічні операції з метою отримання закритих контурів великих об’єктів. Під час виконання завдання був створений алгоритм, за допомогою якого можна автоматично визначати об’єкти, які відрізняються кольором від фону.

Виконав: ІП-14 Бабіч Д. В.