Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

ДОСЛІДЖЕННЯ ІЄРАРХІЧНОЇ СУПЕРПІКСЕЛЬНОЇ СЕГМЕНТАЦІЇ ЗОБРАЖЕНЬ
(тема)

Виконав:
студент 2 курсу, групи ІНФм-21-1
Філатов В.В.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник проф. Машталір В.П.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2022 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2022 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Філатову Владиславу Владиславовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження ієрархічної суперпіксельної сегментації зображень

затверджена наказом університету від 9 листопада 2022 року № 1469Ст

2. Термін подання студентом роботи до екзаменаційної комісії 25 листопада 2022 р

3. Вихідні дані до роботи математичні моделі ієрархічної суперпіксельної сегментації зображень, перелік використовуваних програмних засобів, теоретичні відомості про методи суперпіксельної сегментації зображень.

4. Перелік питань, що потрібно опрацювати в роботі
1. Огляд основних методів суперпіксельної сегментації.
2. Проєктування та розробка застосунку для реалізації математичних моделей ієрархічної суперпіксельної сегментації.
3. Порівняння алгоритмів.
4. Аналіз даних порівняння алгоритмів.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) актуальність проблеми обробки зображень, постановка задачі, тестові зображення, результати роботи методів суперпіксельної сегментації.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Консультант з дотримання діючих стандартів та норм	Доцент Творошенко І.С.		

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	09.11.2022	
2	Аналіз завдання, підбір літератури	09.11.22-11.11.22	
3	Аналіз літератури з досліджуваної проблеми	12.11.22-14.11.22	
4	Аналіз технічних засобів	15.11.22-16.11.22	
5	Розробка методу	17.11.22-19.11.22	
6	Програмна реалізація	20.11.22-21.11.22	
7	Оформлення пояснювальної записки	22.11.22-27.11.22	
8	Перевірка на плагіат	28.12.2022	
9	Рецензування	29.12.2022	
10	Підготовка презентації та доповіді	30.11.22-03.12.22	
11	Занесення роботи в електронний архів	04.12.2022	
12	Попередній захист кваліфікаційної роботи	05.12.2022	

Дата видачі завдання 9 листопада 2022 р.

Студент _____
(підпис)

Керівник роботи _____ проф. Машталір В.П.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 60 с., 3 табл., 37 рис., 1 дод., 37 джерел.

АЛГОРИТМ FH, АЛГОРИТМ SLIC, СУПЕРПІКСЕЛІ, КЛАСТЕРИЗАЦІЯ, СЕГМЕНТАЦІЯ ЗОБРАЖЕНЬ.

Об’єктом дослідження є моделі ієрархічної суперпіксельної сегментації зображень об’єктів.

Метою дослідження є розробка методів, базуючись на використанні алгоритмів суперпіксельної сегментації, який дозволить провести їх порівняльний аналіз.

Використано методи суперпіксельної сегментації зображень, таких як: FH, SLIC. Проведено дослідження методів ієрархічної суперпіксельної сегментації зображень

У результаті дослідження здійснена програмна реалізація методів ієрархічної суперпіксельної кластеризації та досліджено їх.

FH ALGORITHM, SLIC ALGORITHM, SUPER PIXELS, CLUSTERIZATION, SEGMENTATION OF IMAGES.

The object of the research is models of hierarchical superpixel segmentation of object images.

The purpose of the research is to develop methods based on the use of superpixel segmentation algorithms, which will allow for their comparative analysis.

The methods of superpixel segmentation of images, such as: FH, SLIC, were used. A study of hierarchical superpixel image segmentation methods was conducted

As a result of the research, the software implementation of hierarchical superpixel clustering methods was carried out and they were investigated.

ЗМІСТ

Перелік умовних позначень, символів, одиниць, скорочень і термінів 7
Вступ 8
1 Огляд основних методів суперпіксельної сегментації зображень 11
1.1 Визначення ієрархічної сегментації зображень 11
1.2 Приклади застосування сегментації 13
1.3 Особливості задачі сегментації зображень 15
1.4 Алгоритми суперпіксельної сегментації зображень 16
1.5 Оцінка якості алгоритмів сегментації зображень 19
1.6 Постановка задачі дослідження 20
2 Математичні моделі суперпіксельної сегментації зображень 22
2.1 Алгоритми суперпіксельної сегментації зображень 22
2.2 FH (Felzenszwalb and Huttenlocher) 22
2.3 Watershed 25
2.4 Quick Shift 26
2.5 SLIC (Simple Linear Iterative Clustering) 28
2.6 Порівняння даних алгоритмів 30
2.6.1 Порівняння на швидкодію 30
2.6.2 Порівняння стійкості до шуму типу Salt&Pepper 31
2.6.3 Порівняння стійкості до розмиття 32
2.6.4 Порівняння роботи на текстурному зображенні 32
2.6.5 Порівняння роботи на градієнтному зображенні 34
3 Комп’ютерна модель фільтрації зображень 36
3.1 Обґрунтування вибору середовища програмної реалізації 36
3.2 Програмна реалізація 39
3.3 Інструкція користувача 41
3.4 Тестування розробленої моделі 42
3.5 Приклад ієрархічної сегментації 45

3.6 Порівняння алгоритмів 46
3.6.1 Порівняння на швидкодію 46
3.6.2 Порівняння за якістю роботи з границям об’єктів 48
Висновки 52
Перелік джерел посилання 53
Додаток А Тестові зображення 58

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

CPU – Central Processing Unit (центральний процесор)

API – Application Programming Interface (інтерфейс прикладного програмування)

GUI – Graphical User Interface (графічний інтерфейс користувача)

RAG – Region Adjacency Graphs (граф суміжності регіонів)

ВСТУП

Цифрове зображення – графічна форма подання даних, призначена для зорового сприйняття. Кожне 2D зображення передається і запам'ятовується рядково у вигляді одномірного сигналу. Зображення відображаються: на дисплей у векторній або растровій формі.

Комп’ютерний зір – автоматичне отримання з зображення інформації. Під інформацією може бути все, що завгодно: виявлення та розпізнавання об’єктів, групування зображень та пошук за змістом та інше. На практиці комп’ютерний зір представляє собою поєднання програмування, моделювання та математики.

У процесі поліграфічного виробництва усі ілюстрації й елементи оформлення представлені у вигляді цифрових зображеннь різних типів. Цифрові зображення за способом дискретизації оригіналу поділяють на різні типи, такі як: растрові, векторні та змішаного типу.

До растрових зображень відносять двовимірні масиви даних (матриці пікселів). Кожен елемент у цих матрицях представляє з себе ділянку оригіналу з усередненим колірним показником.

Що ж до векторних зображень, то елементами векторного зображення є вектор та крива Безьє. У комп’ютерній графіці вектор розуміється як відрізок, який об’єднує дві точки з заданими координатами. Основним керуючим елементом кривої Безьє є вузол, контрольна точка (control point, CP) або контрольна вершина (control vertex, CV). Ступінь викривлення цієї лінії визначається за допомогою координат вузла а також двох керувальних точок.

Зображення змішаного типу за своєю суттю це поєднання вище зазначених типів. Такі зображення представляють з себе масиви даних, що містять певну інформацію як у вигляді матриці пікселів, так і у вигляді опису векторів, кривих Безьє, примітивів та текстових блоків.

Кластеризація [1-10], яку іноді називають «сегментацією», має на увазі виділення з вхідної множини даних груп об’єктів зі схожими властивостями і часто виступає першим кроком при аналізі даних. Завдання кластеризації масивів багатовимірних спостережень різної природи, метою якої є знаходження в вибірках даних однорідних в прийнятому сенсі груп (сегментів, кластерів, класів) спостережень, а її результати можуть бути використані у безлічі застосунків.

Сегментація зображення - завдання пошуку груп пікселів, кожна з яких характеризує один смисловий об'єкт. У статистиці ця проблема відома як кластерний аналіз і є широко вивченою областю із сотнями різних алгоритмів. У комп'ютерному зір сегментація зображення є однією з найстаріших проблем, що широко вивчаються.

Сегментація [11-17], під час якої зображення розбивається на області, що не перетинаються, називається тесселяцією.

Зараз набуває значного поширення відносно новий підхід до сегментації зображення. Даний підхід заснований на так званій суперпіксельній сегментації. Основою цього підходу є розбиття зображення на множину малих фрагментів зображення, що представляють з себе однорідну групу пікселів, які розташовано поруч один до одного. При подальшій обробці зображення усі пікселі, що потрапляють до складі суперпікселя, сприймаються як єдине ціле. При цьому суперпіксель не обов’язково повинен мати правильну форму. Іншими совами, суперпіксельна сегментація дозволяє зображення, що складається з тисяч, мільйонів або навіть десятків мільйонів пікселів, спростити усього до сотень або ж десятків суперпікселів.

На сьогодні відомо багато різноманітних алгоритмів суперпіксельної сегментації зображення, які використовують різні ознаки схожості та підходи побідови суперпікселів.

Постає нагальне питання оцінки якості роботи та швидкості опрацювання суперпіксельної сегментації зображень різними алгоритмами.

Актуальність дослідження полягає у оцінці якості та швидкості ієрархічної суперпіксельної сегментації різних алгоритмів з метою покращення.

1 ОГЛЯД ОСНОВНИХ МЕТОДІВ СУПЕРПІКСЕЛЬНОЇ СЕГМЕНТАЦІЇ ЗОБРАЖЕНЬ

1.1 Визначення ієрархічної сегментації зображень

Однією з найбільш розв’язуваних задач при обробці зображень є сегментація – розбиття зображення на області, які не перетинаються, що покривають усі зображення і однорідні за деякими ознаками: колір, яскравість, текстура і т.д. На практиці, більша частина базових алгоритмів обробки зображень використовують подання зображеня у вигляді регулярної сітки пікселів, що зумовлено, здебільшого, способом зберігання зображень у цифровій формі. Однак даний варіант не завжди є оптимальним рішенням з багатьох точок зору і насамперед для організації подальшої обробки.

Суперпіксельна сегментація стала важливою низькорівневою задачею в багатьох засобах обробки зображення та компьютерного зору. Суперпікселі просто з’єднують області зображення з перцепційно подібними характеристиками, операції з меншою кількістю примітивів зображень.

Суперпіксельна мапа зображеннь, отримана у результаті виконання суперпіксельної сегментації, має низку значних переваг відносно звичайної регулярної сітки пікселів.

Перш за все, це обчислювальна ефективність: мільйони окремих пікселів замінюються лише на сотні або ж навіть десяки суперпікселів, які виступають в алгоритмах як єдине ціле, що у свою чергу значно знижує час обробки для алгоритмів розпізнавання.

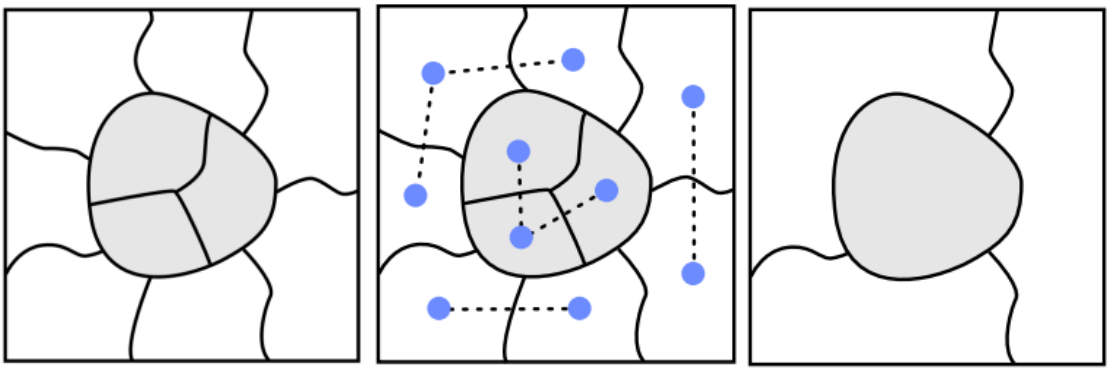
По-друге, у випадку суперпіксельного подання зображення можна говорити про взаємозв’язки між віддаленими один від одного суперпікселями, у той же час у піксельному – лише про зв’язки між пікселями, що лежать безпосередньо поруч.

По-третє, суперпікселі є більш значущими, адже кожен суперпіксель це узгоджена одиниця. Через те що пікселі, які належать суперпікселю, мають схожість за кольором, яскравістю та іншими ознаками. Дані властивості суперпікселів зумовлюють їх доцільність використання у вирішення завдань сегментації об’єктів як із відомими, так і з невідомими властивостями.

З іншого боку, ієрархічний образ сегментація (або ієрархічна сегментація) – це вкладена колекція поступово більш грубих розділів зображення. Вона дає змогу багатомасштабного аналізу зображень із застосуваннями для анотації ієрархічної сцени, виявлення спільної помітності та спільної сегментації.

Класичні алгоритми для ієрархічної сегментації зображень будують ієрархію за допомогою серії операцій, таких як об’єднання регіонів, яке об’єднує дві або більше пов’язаних областей в одну, або розбиття області, яке розділяє область на дві або більше менших з’єднаних областей. За визначенням, кожен крок у цих алгоритмах можна розглядати як нову шкалу сегментації великої кількості шкал спостереження – ієрархічних рівнів, які відрізняються від більшої кількості регіонів (найтонший масштаб) до меншої кількості областей (найбільш грубий масштаб). Таким чином, ми називаємо їх щільними ієрархіями та методами щільної ієрархічної сегментації зображень (або просто щільними методами).

Суперпіксельні алгоритми спочатку були розроблені для створення розділу зображення з набору пікселів. Деякі з цих алгоритмів можна легко виконати рекурсивно для кожного поточного розділу зображення, щоб створити наступний більш грубий розділ зображення для ієрархічної сегментації зображення. Це дозволяє додавати новий більш грубий розділ до ієрархії на кожній ітерації, як правило, зі значно меншою кількістю регіонів, ніж попередній, і загалом створюючи відносно невелика кількість шкал спостереження. Алгоритм ієрархічної сегментації зображення показано на рисунку 1.1.



а) б) в)

Рисунок 1.1 – Алгоритм ієрархічної сегментації зображення: а) перший крок розбиття на суперпікселі; б) об’єднання за схожістю; в) результуюче розбиття

1.2 Приклади застосування сегментації

Однією з основних задач в області комп’ютерного зору є задача отримання інформації і даних із цифрових зображень.

Варто зазначити що сегментація фото та відео контенту досить широко використовується і вже давно присутня у нашому житті. Однак для різних задач використовуються різні методи сегментаціїї, приклад показано на рисунку 1.2. Наведемо декілька прикладів використання методів сегментації:

- медичні зображення;
- виявлення пухлин та інших патологій;
- хірургія за допомогою комп’ютера;
- виділення об’єктів на супутникових знімках;
- розпізнавання осіб;
- обробка супутникових зображень;
- системи керування дорожнім рухом;
- машинний зір.

Рисунок 1.2 – Приклад застосування ієрархічної сегментації зображення

Розглянемо більш детально виділення границь об’єктів на супутниковому знімку із використанням суперпіксельної сегментації як показано на рисунку 1.3.

а) б) в)

Рисунок 1.3 – Формування суперпікселей зображення: а) вхідне зображення; б) сегментація на основі мінімального покривного дерева; в) вирахування карти висот

1.3 Особливості задачі сегментації зображень

При обробці зображень, сегментація зображень розглядається як дуже значущий і важливий процес. Аби отримати будь-яку цінну інформацію із зображення, на ньому варто відокремити регіони з однаковими характеристиками, зокрема, знайти регіони з схожою текстурою чи виділити об’єкти переднього або заднього плану.

Сегментацію зображення ще можна переформулювати як задачу поділу двовимірного сигналу на зв’язкові області (гладкі, у сенсі характеристик аналізу), які відрізняються від сусідніх областей за особливими кольоровими, яскравими або текстурними характеристиками.

Відповідно моделі вхідного сигналу варто зазначити такі особливості сегментації зображення:

– зображення це двовимірний сигнал що відрізняється високою просторовою кореляцією характеристик сусідніх елементів;

– області на зображеннях у більшості випадків мають доволі великі розміри, причому протягом цих областей значення показників можуть сильно змінюватися;

– можлива кількість класів об’єктів на зображенні зазвичай наперед невідома;

– при сегментації зображення, як правило, не ставиться задача об’єднання в один клас близьких за характеристикамиобластей, що не межують між собою.

Ураховуючи вищезазначені особливості, елементи одного і того ж об’єкта зображення в просторі ознак можуть відображатися в досить протяжну область складної форми. У свою чергу, області, що відповідають різним об’єктам, можуть бути досить близькими або навіть перетинатися.

Слід зазначити, що у випадку поелементного відображення втрачається інформація про просторове сусідство точок зображення.

Усе це свідчить тому, що сегментація зображення шляхом розв’язання класичної задачі розпізнавання має доволі вузький спектр застосування. Задовольнити зазначеним особливостям має підхід, який використовує критерії близькості як вибраних ознак, так і просторового розташування елементів зображення. Виходячи з цього, було прийнято ієрархічний підхід, заснований на пірамідальному алгоритмі, який дозволяє порівнювати ознаки просторово сусідніх пар елементів і об’єднувати ті, які в цьому сенсі виявляться близькими.

Отже за цим алгоритмом зображення подається як граф, на нижньому рівні якого розташовані елементи зображення, а гілки відповідають окремим об’єктам. Важливим у цьому є вибір оцінки близькості сусідніх вершин графа, що в застосуванні до розглянутого завдання означатиме вибір простору ознак і спосіб оцінки близькості точок на цьому просторі. Зіставляти вершини графа, відповідні сусіднім елементам або сегментам зображення, зручніше за допомогою відстані на просторі ознак, що і буде критерієм при ухваленні рішення про їх злиття або проведення між ними кордону.

1.4 Алгоритми суперпіксельної сегментації зображень

Для сегментування зображення було розроблено декілька універсальних алгоритмів які наведено на рисунку 1.4.

Так як універсального рішення для задачі сегментації зображень не існує, часто дані методи поєднуються зі знаннями з предметної області, для більш ефективного вирішення поставленого завдання.

Рисунок 1.4 – Загальна класифікація методів сегментування зображень

За однією з класифікацій алгоритмів поділяють існуючі методи на наступні підмножини:

– алгоритми порогової обробки зображень;
– алгоритми виділення країв;
– алгоритми вилучення областей.

Якщо брати алгоритми порогової обробки, то вони є доволі простими в реалізації. Поріг – це властивість (ознака), яка допомагає поділити шуканий сигнал на класи. Операція порогового розподілу полягає у співставленні функції (для прикладу функція повертатиме яскравість пікселя) для кожного пікселя на зображенні із заданим значенням порога.

Алгоритми зазначеної групи можуть давати неточні результати, особливо у випадках наявності шумів на зображенні, що часто вимагає додаткової перед- або пост-обробки зображення.

Найпростіше перетворення – бінаризація. Вона являє з себе операцію порогового поділу, яка в результаті дає бінарне зображення. Мета операції бінаризації полягає у радикальному зменшенні кількості інформації, що міститься на зображенні. У процесі бінаризації вхідне напівтонове зображення, яке має певну кількість ступенів яскравості, перетворюється на чорно-біле (бінарне) зображення, пікселі якого мають тільки два можливих значення 0 і 1 (білий або чорний).

Порогову обробку розподіляють на декілька видів:

– бінаризація з нижнім порогом;
– бінаризація з верхнім порогом;
– неповна порогова обробка;
– багаторівневе граничне перетворення.

Алгоритми виділення країв дозволяють виділити та позначити межі об’єктів чи деяких елементів розміщених на зображенні. Приклад алгоритму виділення країв на зображенні показано на рисунку 1.5.

Рисунок 1.5 – Приклад алгоритму виділення країв

1.5 Оцінка якості алгоритмів сегментації зображень

Якість сегментації зображення може визначатися спектральними, статистичними, характеристиками яскравості зображення. На практиці здебільшого якість розглядається суто як міра близькості двох зображень: ідеального та реального або ж перетвореного та вихідного [18-20]. При такому підході можна оцінювати як суб'єктивний рівень схожості зображень, так і отримати об'єктивні оцінки параметрів сигналів зображення:

– моменти першого та другого порядку різності сигналу порівнюваних зображень;

– параметри перетворення у вигляді відношення сигналу, коефіцієнти стиснення інформації тощо.

Щоб провести порівняльний аналіз методів сегментації використовують критерії, які базуються на обчисленні міри відмінності між результатом сегментації, який отримано за допомогою алгоритму, та сегментом, який побудовано експертом на основі суто візуального аналізу зображення.

Далі наведено критерії оцінки якості сегментації:

– показник надмірної сегментації;

– показник недостатньої сегментації;

– загальна помилка сегментації.

У ході сегментації зображення виникають помилки двох типів:

– на сегментованому зображенні крапка відзначена як контурна, а на ідеальному контурному зображенні вона не відноситься до контуру;

– на сегментованому зображенні точка не позначена як контурна, але вона є такою на ідеальному контурному зображенні.

Тому під час оцінці якості сегментації вибрано два основних критерія: критерій, що показує рівень подібності сегментованого та ідеального контурного зображень (FOM).

Суб’єктивні критерії – критерії візуального сприйняття. Тобто це критерії оцінювані у процесі експертизи деякою групою експертів (спостерігачів). Найбільш поширеним є метод оцінок, у якому експерт оцінює якість зображення за певною шкалою балів, уважаючи, що ідеальне зображення має певний максимальний бал як показано у таблиці 1.1.

Таблиця 1.1 – Дані з оцінками

Нормалізована	Десятибальна		Семибальна шкала погіршення
	Шкала якості	Шкала погіршення	
1	10(відмінно)	10 (непомітно)	1 (помітно)
0,875	-	-	2(ледве помітно)
0,75	8(добре)	8(помітно, не заважає)	3(погіршує, але не заважає)
0,625	-	-	4(погіршує, але не заважає)
0,5	6(задовільно)	6(помітно, трохи заважає)	5(трохи заважає)
0,25	3(не задовільно)	3(заважає)	6(заважає)
0	1(погано)	1(сильно заважає)	7(сильно заважає)

1.6 Постановка задачі дослідження

Виходячи з вищезазначеної інформації, видно що задачу кластеризації зображення можна виконати по різному. Якщо ж брати до уваги найбільш відомі алгоритми суперпіксельної сегментації, то найбільшої популярності, ураховуючи їх відносну простоту реалізації, набули SLIC та FH.

Як вже відомо описані алгоритми мають свої недоліки та переваги. Так, наприклад, найшвидшими та найточнішим при роботі з текстурними зображеннями є алгоритм FH, а от алгоритм SLIC дуже гарно порається із шумними та градієнтними зображеннями. Отже потрібно реалізувати наведені алгоритми та дослідити який з цих алгоритмів покаже себе краще при ієрархічній сегментації.

Отже, сегментація зображення на суперпікселі є актуальним завданням обробки та розпізнавання зображень. Використовуючи суперпіксельну сегментації можна значно поліпшити процес обробки зображення. Через це ставиться завдання порівняння найбільш відомих алгоритмів суперпіксельної сегментаціїї.

Об’єктом дослідження є моделі ієрархічної суперпіксельної сегментації зображень об’єктів.

Метою дослідження є розробка методів, базуючись на використанні алгоритмів суперпіксельної сегментації, який дозволить провести їх порівняльний аналіз.

Для досягнення мети необхідно вирішити наступні завдання:

– провести аналіз існуючих алгоритмів;

– розробити застосунок, що реалізує методи суперпіксельної сегментації;

– провести порівняльний аналіз алгоритмів при різних умовах;

– дослідити, отримані під час аналізу, дані.

2 МАТЕМАТИЧНІ МОДЕЛІ СУПЕРПІКСЕЛЬНОЇ СЕГМЕНТАЦІЇ ЗОБРАЖЕНЬ

2.1 Алгоритми суперпіксельної сегментації зображень

Далі описано декілька популярних алгоритмів генерації суперпікселів та проведено їх невеликий порівняльний аналіз. Нижче наведено приклад схеми обробки пікселів при сегментації як показано на рисунку 2.1.

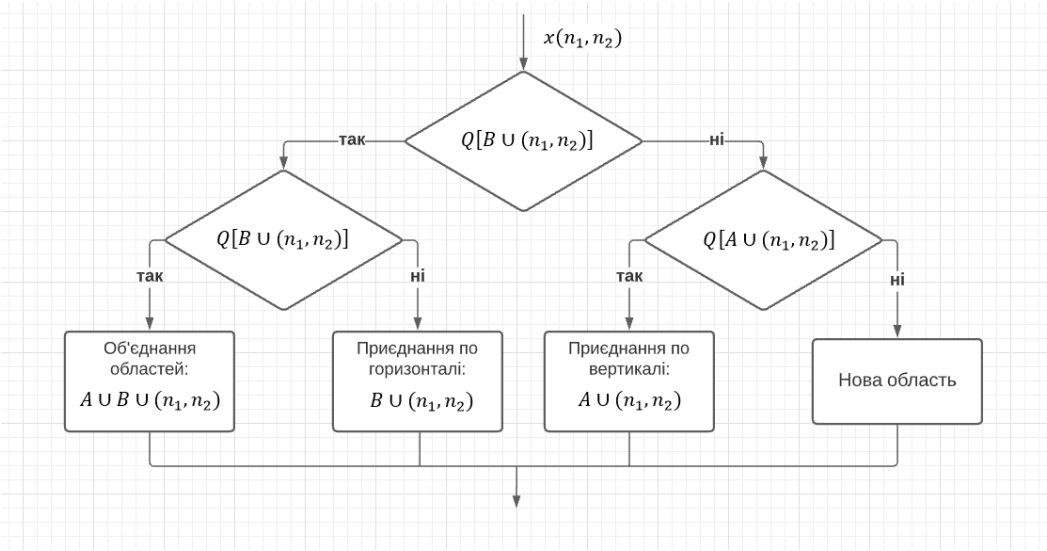


Рисунок 2.1 – Схема обробки окремих пікселів при сегментації зображень

2.2 FH (Felzenszwalb and Huttenlocher)

FH – алгоритм, який базується на представленні зображення у вигляді графа [21]. Вага ребер графа, що поєднують сусідні вершини (пікселі), обчислюється як різниця інтенсивності:

$$w((v_i, v_j)) = |I(p_i) - (p_j)|. \quad (2.1)$$

Основна ідея полягає у побудові сукупностей зв'язних вершин (кластерів), щоб у середині кластеру вага ребер була меншою у порівнянні із вагою ребер, які поєднують вершини різних кластерів. Виходячи зі зазначеного вище, вводяться поняття внутрішньої різності $Int(C)$ та різності між двома компонентами $Dif(C1, C2)$.

$$Int(C) = \max_{e \in MST(C,E)} w(e), \quad (2.2)$$

де $w(e)$ – вага ребра e;

$MST(C, E)$ – мінімальне остовне дерево компонента C.

$$Dif(\mathrm{C1}, \mathrm{C2}) = \min_{v_i \in C_1, v_j \in C_2, (v_i, v_j) \in E} w((v_i, v_j)), \quad (2.3)$$

де (v_i, v_j) – ребро, що поєднує вершини v_i, v_j.

Далі вводиться предикат наявності границь між кластерами $D(C1, C2)$:

$$D(C1, C2) = \begin{cases} true\ if\ Dif(C1, C2) > MInt(C1, C2) \\ false\ otherwise \end{cases}, \quad (2.4)$$

де мінімальна внутрішня різність $MInt$ визначається за формулою:

$$MInt(C1, C2) = \min(Int(C1) + \tau(C1), Int(C2) + \tau(C2)), \quad (2.5)$$

де τ – порогова функція, що контролює на скільки різниця між двома компонентами повинна бути більшою у порівнянні з їх внутрішніми різностями, аби провести між ними границю.

Задля зменшення кількості помилок у невеликих компонентах, використовується наступна формула:

$$\tau(C) = k/|C|, \tag{2.6}$$

де $|C|$ визначає розмір компонента C;

k – константа, яку задаватиме користувач.

Коефіцієнт k дозволяє опосередковано впливати на розмір результуючих суперпікселів. Фактичний розмір і кількість сегментів можуть відрізнятися залежно від локальної контрастності.

Далі на рисунку 2.2 наведено приклад роботи алгоритму.

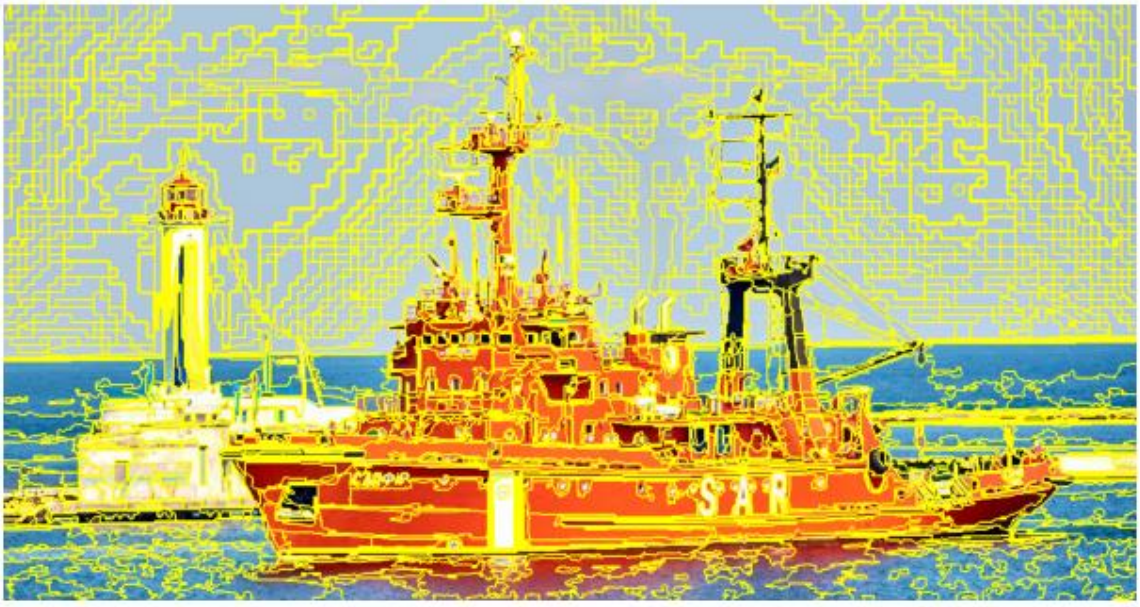


Рисунок 2.2 – Приклад роботи алгоритму FH

2.3 Watershed

Watershed – алгоритм, який базується на ідеї використання «топографічного рельєфу». Ідея полягає у представленні зображення у вигляді топографічного рельєфу, на якому значення кожного елементу зображення описує його висоту в заданій точці. Як правило значення висоти на зображенні відповідає значенню градієнта. Аналогічно фізиці у природі, спершу вода заповнює найнижчі зони, поступово підвищуючи рівень та утворюючи водозбірники. У місцях, де «вода з різних басейнів об’єднується», утворюється границя водорозділу. Отже зображенні ця границя буде представляти з себе границю вихідного кластеру.

Є досить багато способів реалізації даного алгоритму. Розглянемо варіант, запропонований Ф. Мейером. Для відтворення описаного вище процесу використовується впорядкована множина черг з пріоритетом. Вказаний алгоритм розпочинає з виявлення точок локального мінімуму, присвоює їм унікальні відмітки та заносить іх до черги. Далі з черги видаляється елемент з найвищим пріоритетом. Його сусідам, що не мають відміток, присвоюється відмітка видаленого елемента та виконується їх занесення до черги в залежності від пріоритету. Описаний крок повторюється доти, доки в черзі знаходяться елементи. Нижче на рисунку 2.3 наведено приклад роботи алгоритму.

Рисунок 2.3 – Приклад роботи алгоритму Watershed

Compact Watershed – розширення алгоритму Watershed, запропонованого Ф. Мейером, для генерації суперпікселів, які мають більш однорідний розмір та форму.

Зазначений алгоритм відрізняється від свого попередника розподіленням початкових елементів з позначками на регулярній сітці, а не в зонах локального мінімуму. Це й призводить до більш однорідного розподілення результуючих сегментів. Зокрема, тут вводиться обмеження компактності, завдяки заміні критерію пріоритетності в черзі, що дозволяє контролювати форму і розмір результуючих суперпікселів. Нижче на рисунку 2.4 наведено приклад роботи алгоритму Compact Watershed.

Рисунок 2.4 – Приклад роботи алгоритму Watershed

2.4 Quick Shift

Quick Shift – відносно новий алгоритм сегментації 2D-зображень що відноситься до сімейства алгоритмів пошуку моди. Quick Shift засновано на апроксимації ядерного зсуву середнього (mean-shift). Алгоритм зсуває кожну точку x_i до найближчого з сусідів, у якого щільність P більша, утворюючи таким чином траєкторію y:

$$y_i(1) = \underset{j:P_j>P_i}{\operatorname{argmin}} D_{ij}, \tag{2.7}$$

$$P_i = \frac{1}{N}\sum_{j=1}^{N} \phi(D_{ij}). \tag{2.8}$$

Даний алгоритм об’єднує всі точки в одне дерево, у якому ребра, що перевищують порогове значення τ розбиваються, у результаті чого утворюючи кластери. Розрахування щільності може бути по-різному, одним із багатьох варіантів є використання ізотропного ядра Гаусса:

$$P(x) = \frac{1}{2\pi\sigma^2 N}\sum_{i=1}^{N} e^{\frac{-\|x-x_i\|^2}{2\sigma^2}}. \tag{2.9}$$

Основною перевагою алгоритму Quick Shift є те, що він фактично обчислює ієрархічну сегментацію одночасно у кількох масштабах.

Далі на рисунку 2.5 наведено приклад роботи алгоритму.

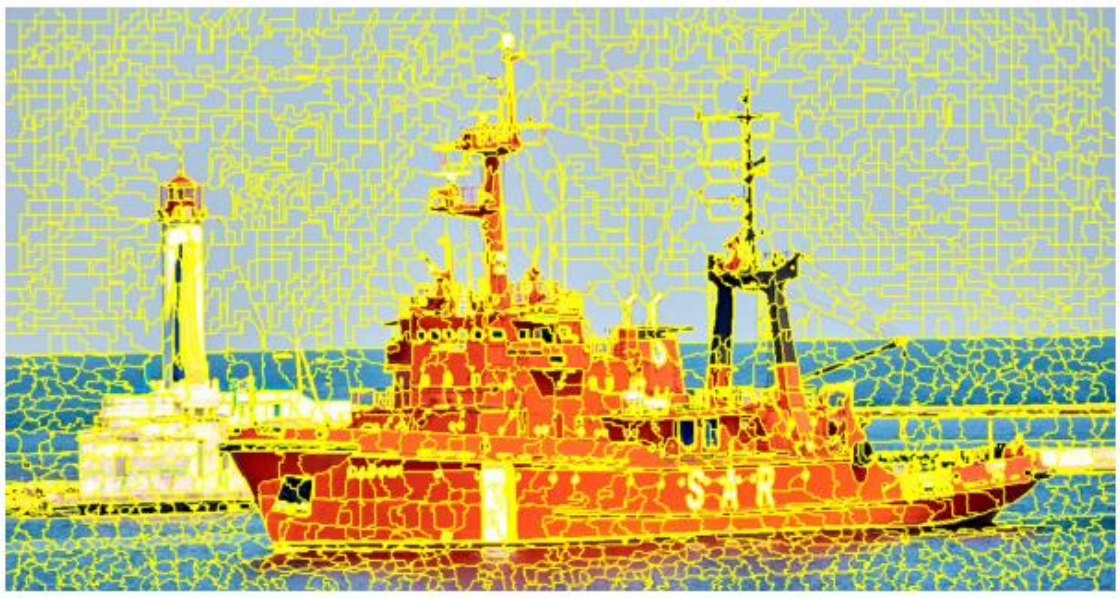


Рисунок 2.5 – Приклад роботи алгоритму Quickshift

2.5 SLIC (Simple Linear Iterative Clustering)

Даний алгоритм генерує суперпікселі на основі схожості пікселів за кольором та близькості на площині зображення. Власно кажучи SLIC [21-23] є покращеним методом кластеризації *k*-середніх. Підвищення швидкості у порівнянні із методом *k*-середніх полягає у зменшенні кількості розрахунків дистанції. Це відбувається за рахунок зменшення регіону пошуку для кожного центра кластера.

$$D = \sqrt{d_c^2 + (\frac{d_s}{s})^2 m^2}, \tag{2.10}$$

$$d_c = \sqrt{(l_j - l_i)^2 + (a_j - a_i)^2 + (b_j - b_i)^2}, \tag{2.11}$$

$$d_s = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}, \tag{2.12}$$

де d_c – колірна дистанція;

d_s – просторова дистанція;

m – константа, що дозволяє задавати важливість між колірною схожістю та просторовою близькістю.

Якщо значення m завелике, просторова близькість відіграє більшу роль і, як результат, отримуємо компактні суперпікселі. Тобто, коли значення m замале, стає важливішою колірна схожість та вихідні суперпікселі краще дотримуються меж об’єктів. Однакпри цьому вони мають менш регулярну форму та розмір.

Важливо, щоб цей алгоритм працював у колірному просторі Lab для отримання добрих результатів. Алгоритм швидко набрав обертів і зараз широко використовується.

Далі на рисунку 2.6 наведено приклад роботи алгоритму.

Рисунок 2.6 – Приклад роботи алгоритму

SLICO – версія алгоритму SLIC без додаткових параметрів, що регулюють компактність. У алгоритмі SLICO фактор компактності для кожного кластера розраховується окремо. Іншими словами константа, що дозволяє задавати важливість між колірною схожістю та просторовою близькістю, обирається як максимальна колірна відстань у межі кластеру, що отримана при попередній ітерації. На рисунку 2.7 наведено приклад роботи алгоритму.

Рисунок 2.7 – Приклад роботи алгоритму SLICO

2.6 Порівняння даних алгоритмів

2.6.1 Порівняння на швидкодію

У даному порівнянні використано набір зображень із різним розміром та співвідношенням сторін. Виходячи з того, що кожен алгоритм розбив зображення на різну кількість кластерів, вирахуємо час витрачений алгоритмами на кожні 100 кластерів. Отримані дані занесемо до таблиці 2.1.

Таблиця 2.1 – Дані про час виконання

Дослідження №	Алгоритм			
	FH	SLIC	Quickshift	Watershed
1	0,13	6,2	0,23	3,72
2	0,1	4,46	0,21	3,72
3	0,08	5	0,19	3,740

Для більшої наочності побудуємо графік (рис. 2.8), на основі отриманих даних з таблиці 2.1.

Дивлячись на графік можна сказати, що найшвидшими серед досліджуваних алгоритмів є FH та Quickshift, а найповільнішим серед усіх алгоритмів – SLIC. Також слід відмітити, що алгоритми FH та Quickshift мають тенденцію на скорочення часу опрацювання кожних 100 суперпікселів при обробці малих зображень. Алгоритм Watershed має середній показник швидкості виконання та, виходячи із отриманих даних, байдужий до розмірів зображення, оскільки витрачає однаковий час на побудову 100 сегментів. Алгоритм SLIC виявився найповільнішим за усіх та має непостійну швидкість роботи при обробці зображення.

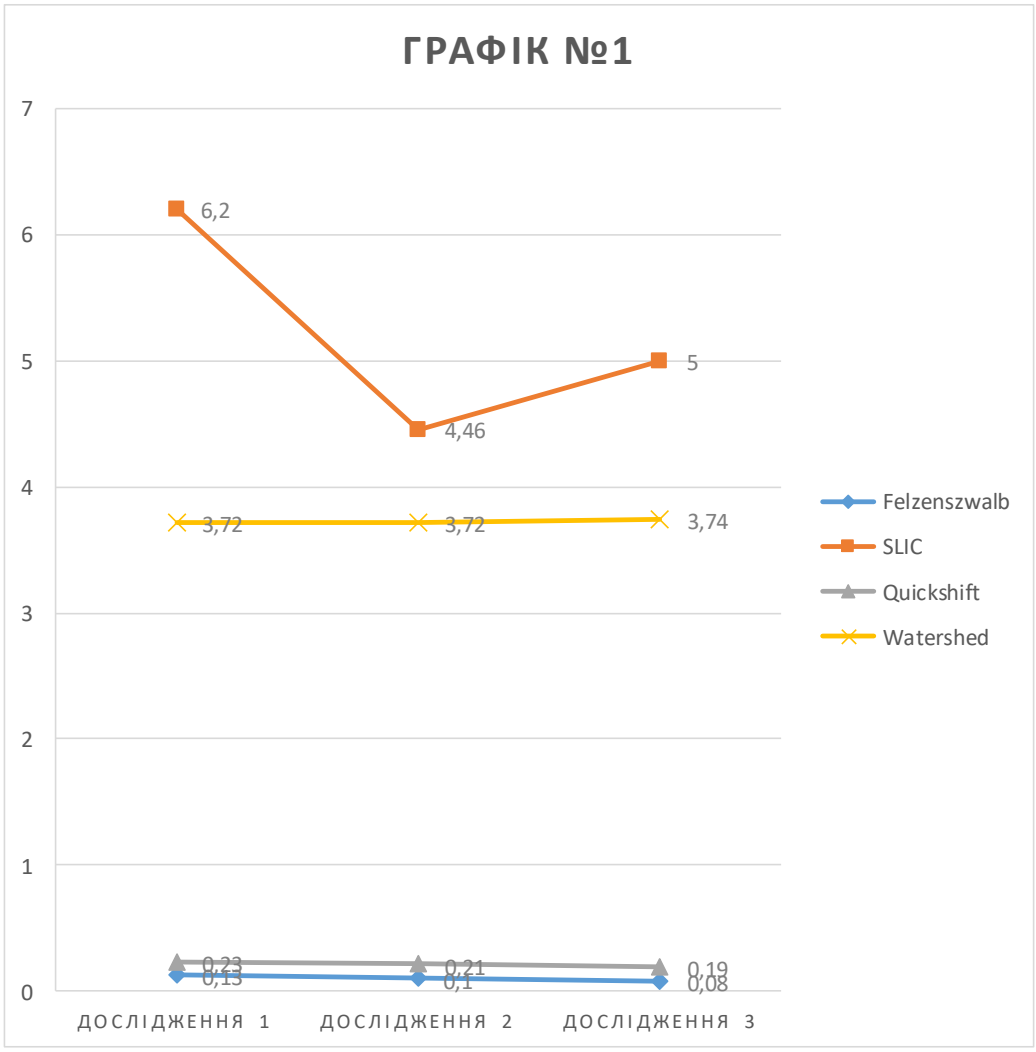


Рисунок 2.8 – Графік порівняння на швидкодію

2.6.2 Порівняння стійкості до шуму типу Salt&Pepper

Слід звернути увагу на актуальність даного порівняння, оскільки даний сценарій моделює як низькоякісні зображень з мережі Інтернет, так і нічні знімки [24-27]. Це відбувається тому що при фотографуванні вночі на матрицю камери потрапляє набагато менша кількість світла. Саме тому виникають шуми на зображенні.

Якщо відкинути витрачений час на опрацювання алгоритму SLIC, то він виявляється найбільш придатним для обробки шумних зображень. Шукаючи компроміс, можна звернути увагу на алгоритм FH. А от алгоритм Quickshift, не зважаючи на швидкість, не підходить для поставленої задачі.

2.6.3 Порівняння стійкості до розмиття

Дане порівняння також має велике значення, так як даний сценарій моделює не лише використання низькоякісних зображень з мережі Інтернет, а й художніх фото з ефектом боке, які зараз широко розровсюджені, наприклад, при портретних зйомках а також при зйомках об’єктів на передньому плані. Боке — розмиття зображення у фокусі об’єктивом у фотографії. Термін, що з’явився в російській мові наприкінці 1990-х років і описує суб’єктивні художні переваги частини зображення, що не у фокусі на фотографії. На багатьох зображеннях фон розмивається фотографом навмисно для візуального виділення головного об’єкта зйомки.

При незначному розмитті зображення кожен алгоритм, окрім Watershed, відпрацював задовільно. Однак, у випадку сильного розмиття, алгоритм FH починає втрачати багато деталей. Виходячи з цього досліду, більш збалансованими є алгоритми SLIC та Quickshift.

2.6.4 Порівняння роботи на текстурному зображенні

Для проведення наступного досліду візьмемо зображення на якому знаходиться кілька областей із різними текстурами. Наприклад, рисунок 2.9.

Рисунок 2.9 – Тестове зображення

Для наочності збільшимо довільну область вихідного та розглянемо її більш детально. Далі на рисунку 2.10 наведено приклад роботи алгоритмів.

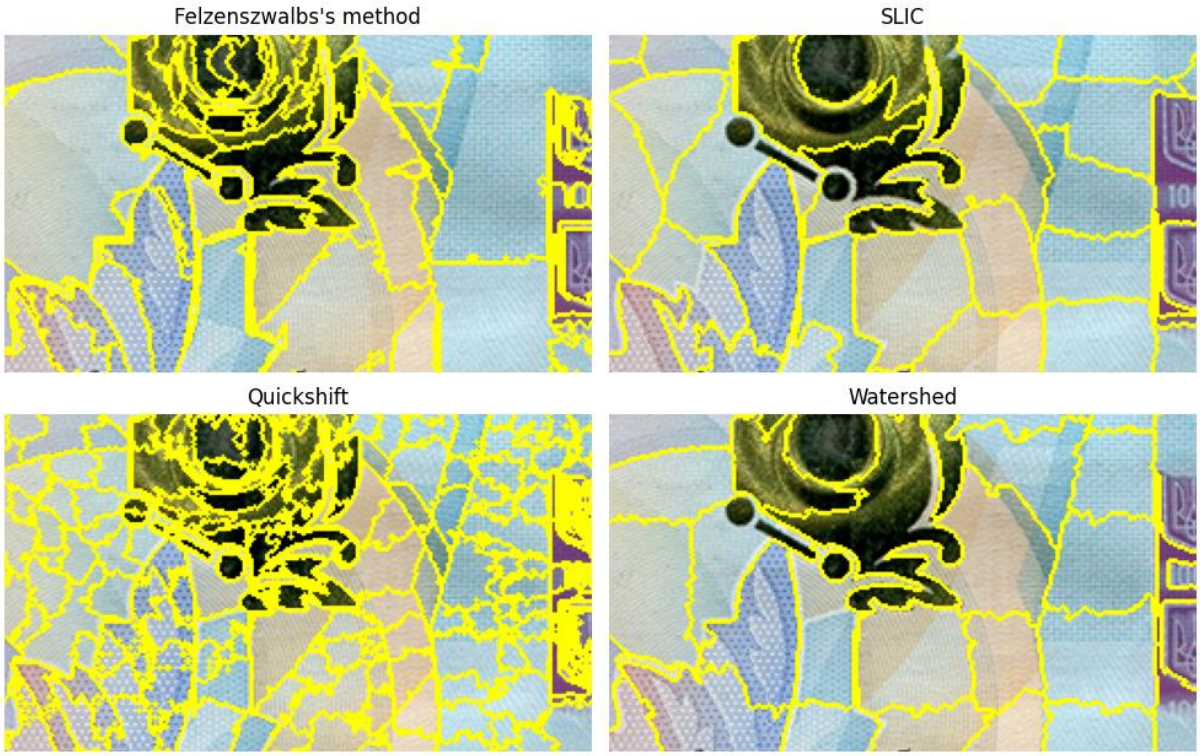


Рисунок 2.10 – Суперпіксельна сегментація текстурного зображення

Дивлячись на рисунок 2.10, можна відмітити що алгоритм FH упорався із завданням виділення окремих текстур найкраще. Трохи гірше але не погано відпрацював алгоритм SLIC. У той же час інші два алгоритми сегментації не впорались із поставленою задачею.

2.6.5 Порівняння роботи на градієнтному зображенні

Для проведення наступного досліду візьмемо зображення з RGB градієнтом як показано на рисунку 2.11.

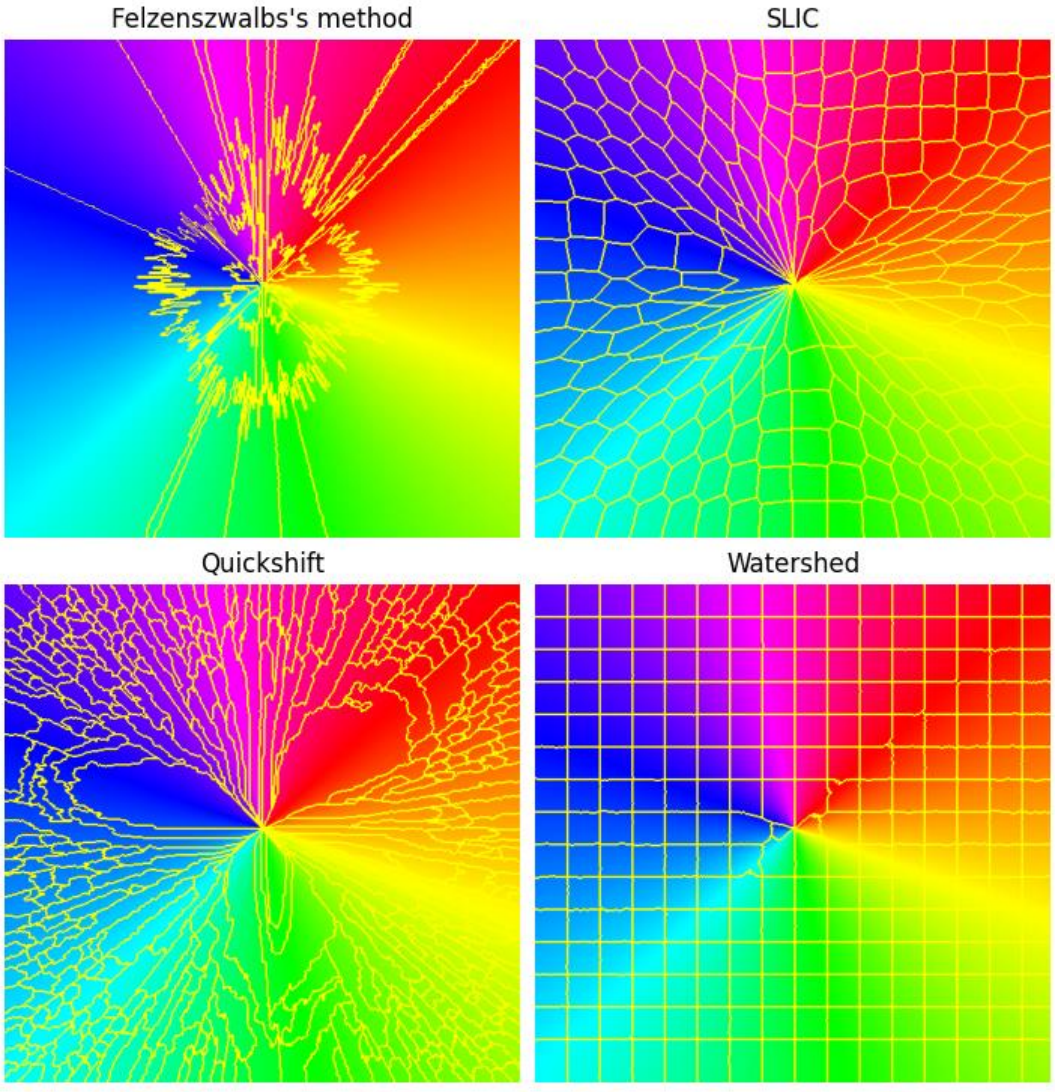


Рисунок 2.11 – Суперпіксельна сегментація текстурного зображення

Розглядаючи рисунок 2.11, варто відмітити те, що алгоритм Watershed повністю провалив завдання. У той же час алгоритм SLIC добре впорався, що й не дивно, оскільки алгоритм базується на колірній схожості.

Розглянувши результати усіх проведених досліджень, можна зазначити те, що кожен алгоритм суперпіксельної сегментації кращий у своєму конкретному сценарії використання. Однак алгоритм Watershed показав себе найгірше за всіх у кожному з поставлених експериментів.

Якщо звертати увагу лише на швидкість алгоритму, то лідером буде алгоритм FH. З досить невеликим відривом від алгоритму FH знаходиться Quickshift.

Якщо ж потрібен алгоритм який краще справляється з зображенням на якому присутні шуми, то варто звернути увагу на SLIC. Менш ефективна, але набагато швидша альтернатива – FH.

У випадку, коли потрібно сегментувати повністю або частково розмите зображення, краще обрати алгоритм SLIC або Quickshift.

У випаду, коли задача полягає у виділенні на зображенні областей з різною текстурою або просто текстурованому зображенні, найкращим варіантом для даної задачі буде використання алгоритму FH.

А, коли потрібно опрацювати градієнтне зображення, варто звернути увагу на алгоритм SLIC, оскільки він базується на колірній схожості.

Але, якщо все ж таки розглядати серед них більш універсальний алгоритм суперпіксельної сегментації, то перш за все варто звернути увагу на SLIC чи FH.

3 КОМП'ЮТЕРНА МОДЕЛЬ ФІЛЬТРАЦІЇ ЗОБРАЖЕНЬ

3.1 Обґрунтування вибору середовища програмної реалізації

У рамках кваліфікаційної роботи було розроблено алгоритми для сегментації зображень за допомогою мови Python.

Операційна система Windows надає розробникам різноманітних застосунків потужні засоби Інтерфейсу Графічних Пристроїв GDI (Graphics Device Interface) для побудови графічних зображень незалежно від типу використовуваного пристрою виводу. Однпк, GDI обтяжує програмістів безліччю додаткових операцій (зокрема, по керуванню системними ресурсами), які відволікають розробника від його основного завдання – створення графіки.

Розробники використовують Python у багатьох місцях. Його багата базова бібліотека робить його чудовим для всіх видів маленьких допоміжних сценаріїв. Але він так само добре масштабується для великих систем. Для ілюстрації: перші творці YouTube здебільшого використовували Python. Як відомо, Dropbox в основному написаний на Python а також весь сервер і веб-сайт Instagram також написані на Python.

Python можна використовувати для автоматизації завдань, виконання обчислень, створення інтерфейсів користувача, створення веб-сайтів, доступу до баз даних, завантаження інформації з Інтернету тощо. Це універсальна мова, яку легко вивчати та писати на ній, і хоча вона ідеальна для програмістів-початківців, такий же корисний і потужний для досвідчених професіоналів.

Python надзвичайно популярний у галузі обробки даних, яка швидко розвивається. Багато дослідників обробки даних використовують Python для повсякденної роботи. І це лише декілька прикладів.

Однією з найбільш помітних особливостей Python є те, як він забезпечує використання відступів для зручності читання.

Для багатьох мов потрібна ручна компіляція, перш ніж буде можна запустити програму, тоді як інші мови є так званими інтерпретованими мовами: їх може запускати безпосередньо інтерпретатор. Python знаходиться десь посередині між цими двома світами.

Під час створення програм на Python, ви можете запускати її безпосередньо без етапу компіляції вручну. Це стосується всіх інтерпретованих мов, і тому більшість вважає що це інтерпретована мова. Однак внутрішньо Python компілює код у код нижчого рівня, який називається байт-кодом. Цей код не є специфічним для архітектури системи (наприклад, Intel проти ARM), але його швидше читати та аналізувати, ніж файли звичайного тексту.

У деяких ситуаціях цей байт-код зберігається на диску у файлах із розширенням *.pyc. Але коли Python використовується як мова сценаріїв, він не буде кешувати байт-код. Можна запускати код безпосередньо, не турбуючись про байт-код, оскільки Python обробляє все це автоматично.

Можна написати код у текстовому редакторі та виконати його безпосередньо. Жодних додаткових кроків, таких як компіляція та зв’язування, не потрібно.

Оскільки це звичайний текст, можна просто відкрити програму та переглянути її вміст. Навпаки, скомпільований код не читається людиною. У такому разі доведеться шукати вихідний код (якщо він взагалі доступний).

Працездатність не залежить від платформи. Код працюватиметься доти, доки на платформі є інтерпретатор Python. Зкомпільований код часто прив’язується до певної платформи, як-от Windows і Linux, і до певної архітектури процесора, як-от Intel або ARM (якщо він не компілюється до проміжного байт-коду, як-от Java і .NET).

Деякі з цих переваг також можуть бути недоліками. Як уже згадувалося, інтерпретовані мови не є високопродуктивними мовами. Крім того, той факт, що вихідний код легко читати та змінювати, не є перевагою для розробників, які хочуть захистити свої авторські права.

Scikit-image – Python-бібліотека з відкритим кодом, яка працює з масивами NumPy. Scikit-image реалізує утиліти і алгоритми для використання в дослідницьких, медичних, освітніх і промислових цілях. Зазначена бібліотека містить високоякісний і рецензований код, написаний активною спільнотою.

OpenCV – бібліотека алгоритмів та функцій з відкритим кодом. Дана бібліотека призначена для обробки зображень і алгоритмів загального призначення на основі комп'ютерного зору [28-30].

EasyGUI – модуль для простого кодування графічного інтерфейсу мовою Python. EasyGUI відрізняється від інших генераторів графічного інтерфейсу тим, що ця бібліотека не керується подіями. Тим часом усі взаємодії із графічним інтерфейсом генеруються простими викликами функцій. EasyGui забезпечує побудову простого у використанні інтерфейс для зручної графічної взаємодії з користувачем.

Atom – один із найвідоміших редакторів коду. Серед іншого, його модульність, його надбудови та його інтеграція з сервісом Github, сервісом репозиторію, досить популярним серед розробників Web застосунків та програмістів. Приклад інтерфейсу зображено на рисунку 3.1.

На момент виходу Atom-IDE включає інтелектуальне автодоповнення синтаксису, навігацію за кодом, перехід до визначення функцій і класів, пошук посилань, інтерактивні підказки, виділення синтаксичних помилок, форматування коду та багато іншого. IDE підтримував TypeScript, Flow, JavaScript, Java, C#, PHP, Rust, Go, Python [31-36] та засоби для глибокого синтаксичного аналізу коду проєктів.

Усі компоненти підтримки мов оформлені у вигляді окремих серверних обробників. Взаємодія з ними здійснюється за допомогою протоколу LSP (Language Server Protocol).

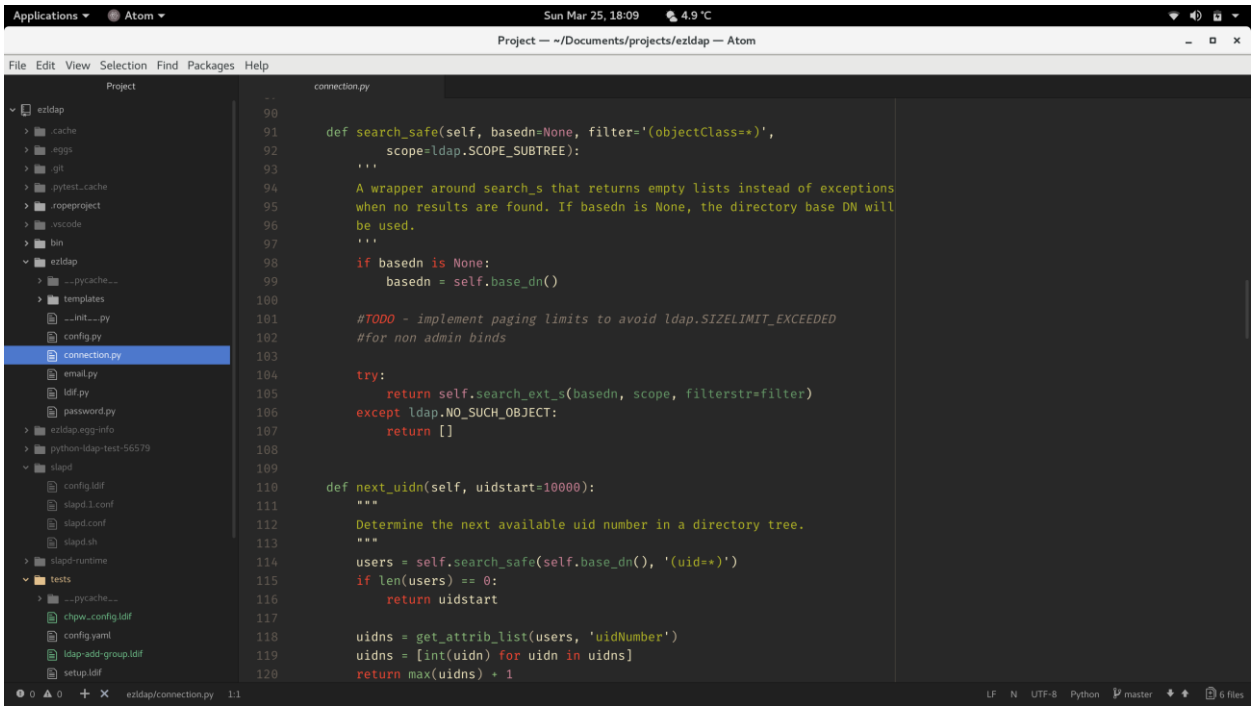


Рисунок 3.1 – Приклад інтерфейсу Atom

3.2 Програмна реалізація

Для програмної реалізації обрано дослідження алгоритму SLIC та FH, мову програмування Python з використанням модулів Scikit-image та EasyGUI.

У цьому розділі описано інструкцію із встановлення мови Python та початку розробки.

Для встановлення Python необхідно завантажити файл установник з офіційного сайту. На наступному кроці виконуємо завантажений файл із офіційного сайту.

Далі варто зробити перевірку чи успішно встановлено Python. Для цього відкриваємо командний рядок та вписуємо команду «`pip`». У результаті отримуємо інформацію як зображено на рисунку 3.2.

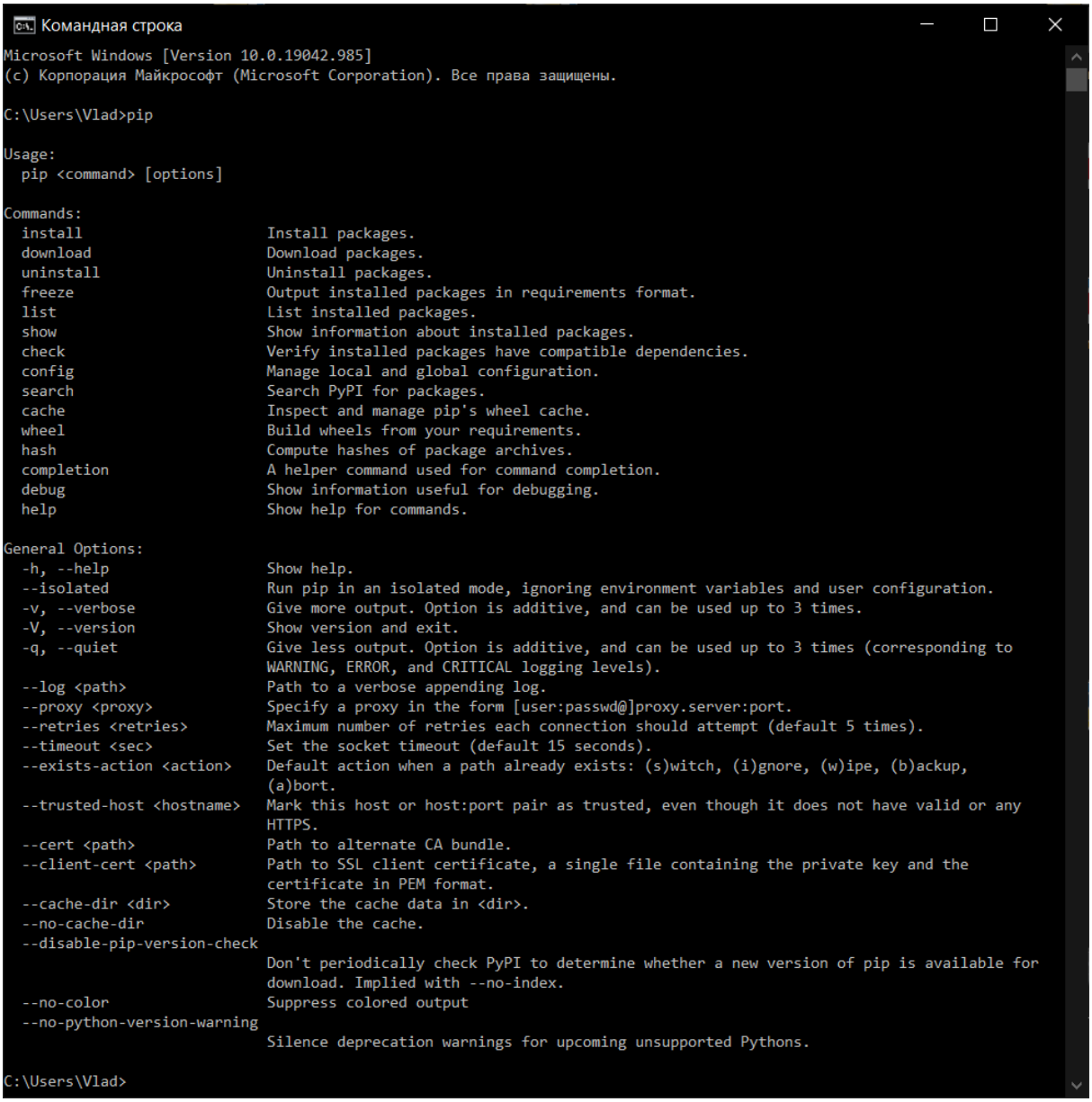


Рисунок 3.2 – Перевірка встановлення Python

Нижче описано процес підключення зазначених раніше бібліотек до проєкту за допомогою командного рядка.

Спочатку відкриваємо командний рядок. Далі вписуємо команду «`pip install opencv-python`», за допомогою якої встановлюємо бібліотеку OpenCV, після чого розпочнеться завантаження та встановлення бібліотеки.

Так само, за допомогою команд «`pip install scikit-image`» та «`pip install easygui`» встановлюємо бібліотеки Scikit-image та EasyGUI відповідно. Нижче приклад встановлення бібліотеки Scikit-image на рисунку 3.3.

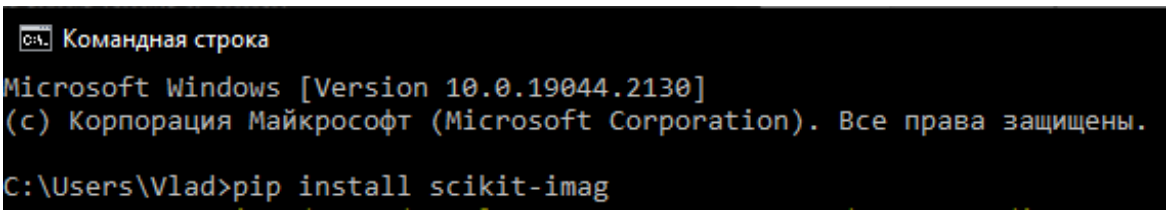


Рисунок 3.3 – Команда встановлення бібліотеки Scikit-image

Далі потрібно створити файл у якому буде зберігатися код програми. Для цього можна скористатися функціоналом IDE або ж просто створити файл у потрібній директорії із припискою «.py».

3.3 Інструкція користувача

Нижче наведено покрокову інструкцію для користувача аби полегшити взаємодію із застосунком:

– виконати застосунок;

– із запропонованих двох варіантів головного меню обрати:

1) «Choose» – обрати зображення;
2) «Exit» – завершення роботи;

У випадку обрання «Choose», користувачу потрібно вказати бажане зображення у новому вікні провідника та натиснути на кнопку «ОК». Наступним кроком відкриється основне вікно застосунку;

– на екрані з вихідними зображеннями результатів обробки можна скористатися іконкою «лупи» для збільшення бажаної області зображення, а також зберегти отримане зображення у потрібне місце на диску, скориставшись кнопкою із зображенням дискети;

– у наступному вікні користувачу подано дані про роботу алгоритмів. Для завершення роботи з застосунком користувач має натиснути хрестик у правому верхньому куті вікна застосунку або просто натиснути на кнопку «ОК».

3.4 Тестування розробленої моделі

Нижче, на рисунках 3.4 – 3.9 наведені зображення принципу взаємодії користувача із застосунком (ієрархічна суперпіксельна сегментація зображення).

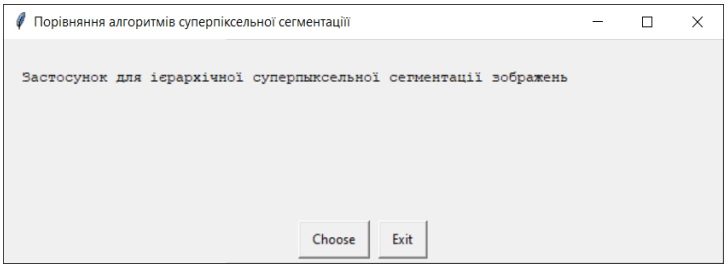


Рисунок 3.4 – Початкове вікно застосунку

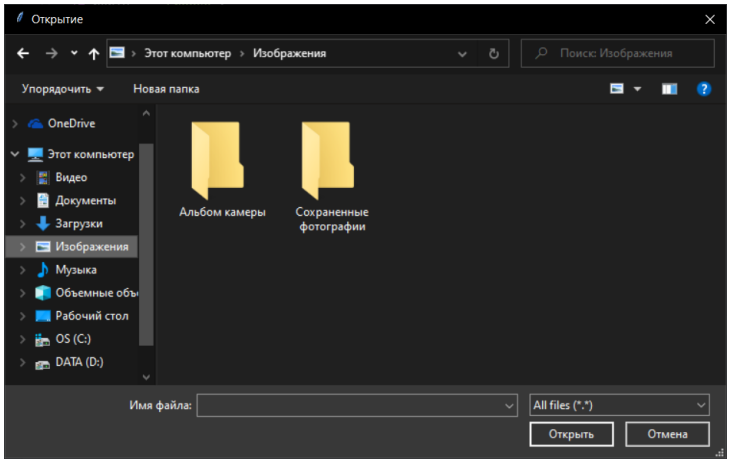


Рисунок 3.5 – Вікно вибору бажаних даних для тесту

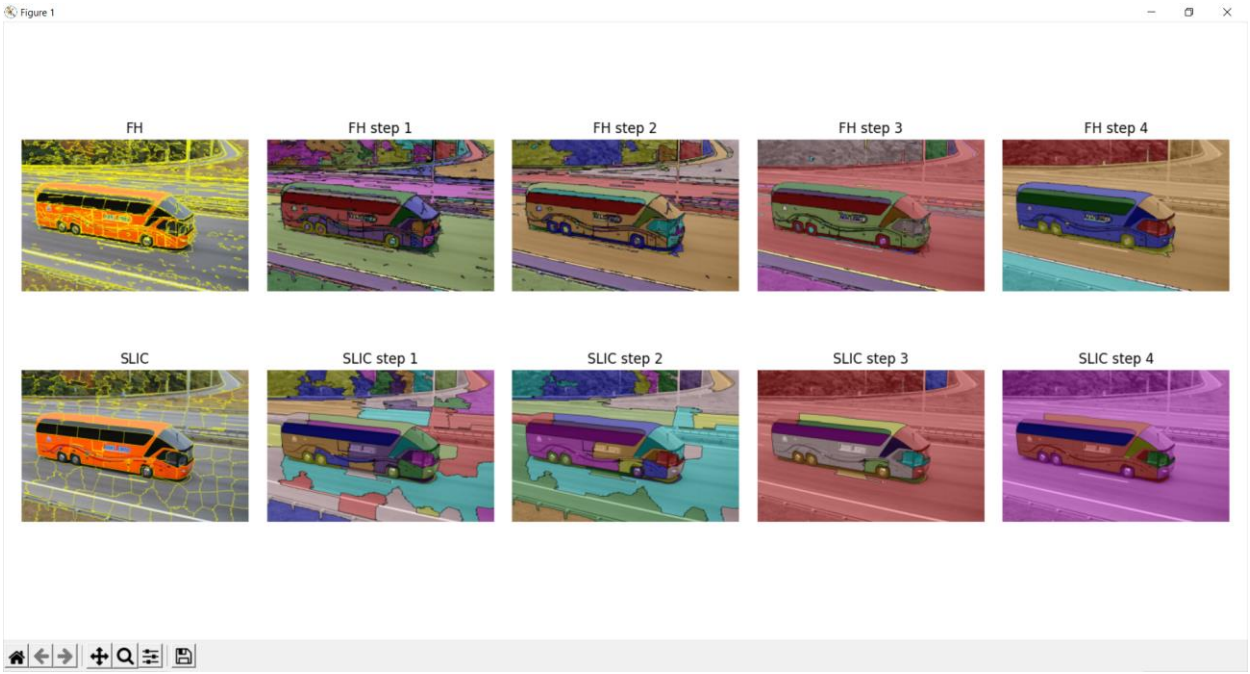


Рисунок 3.6 – Вікно із результатами

Рисунок 3.7 – Вікно із результатами використавши збільшення необхідної області

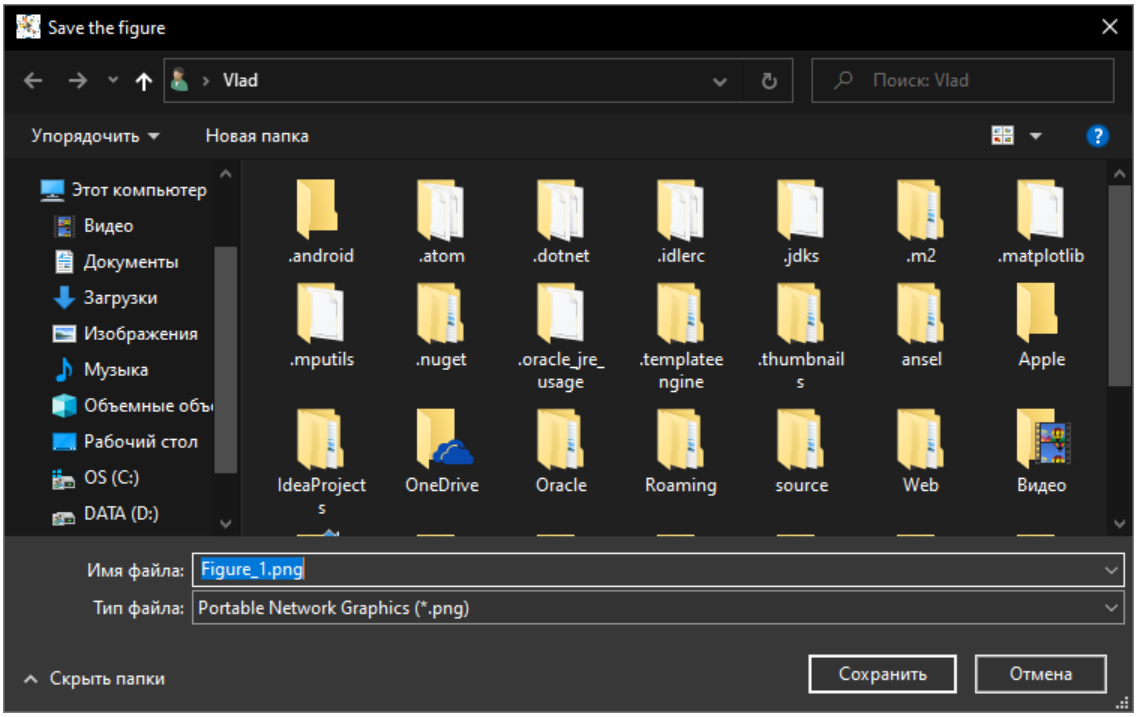


Рисунок 3.8 – Вікно провідника для збереження результатів

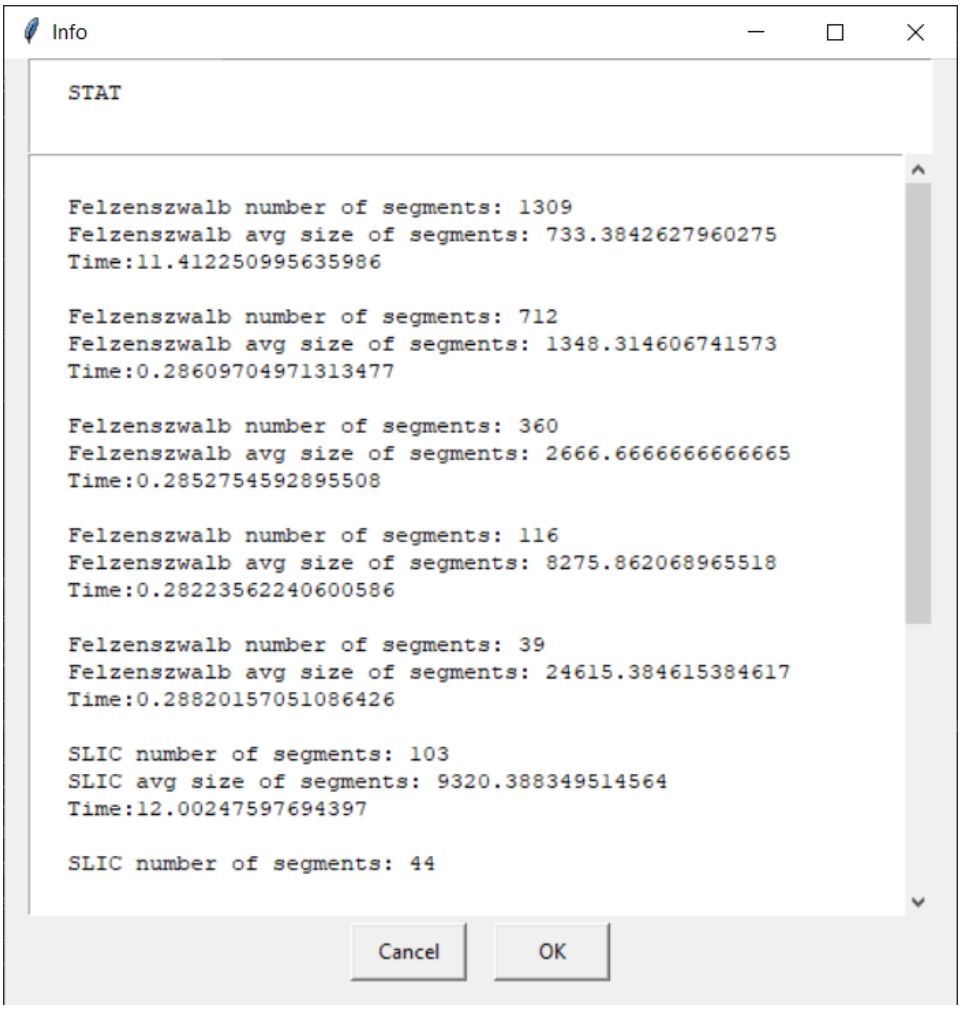


Рисунок 3.9 – Вікно зі статистикою роботи алгоритмів

3.5 Приклад ієрархічної сегментації

В цьому прикладі показано, як виконується ієрархічне складання на графах спільності регіонів на межі регіону (RAGs). Області з найменшими вагами ребер послідовно об’єднуються до тих пір, поки не залишиться ні одного ребра з вагою менше порога. Ієрархічне складання здійснюється за допомогою функції skimage.future.graph.merge_hierarchical() модуля Scikit-image. Приклад роботи показано на рисунку 3.10.

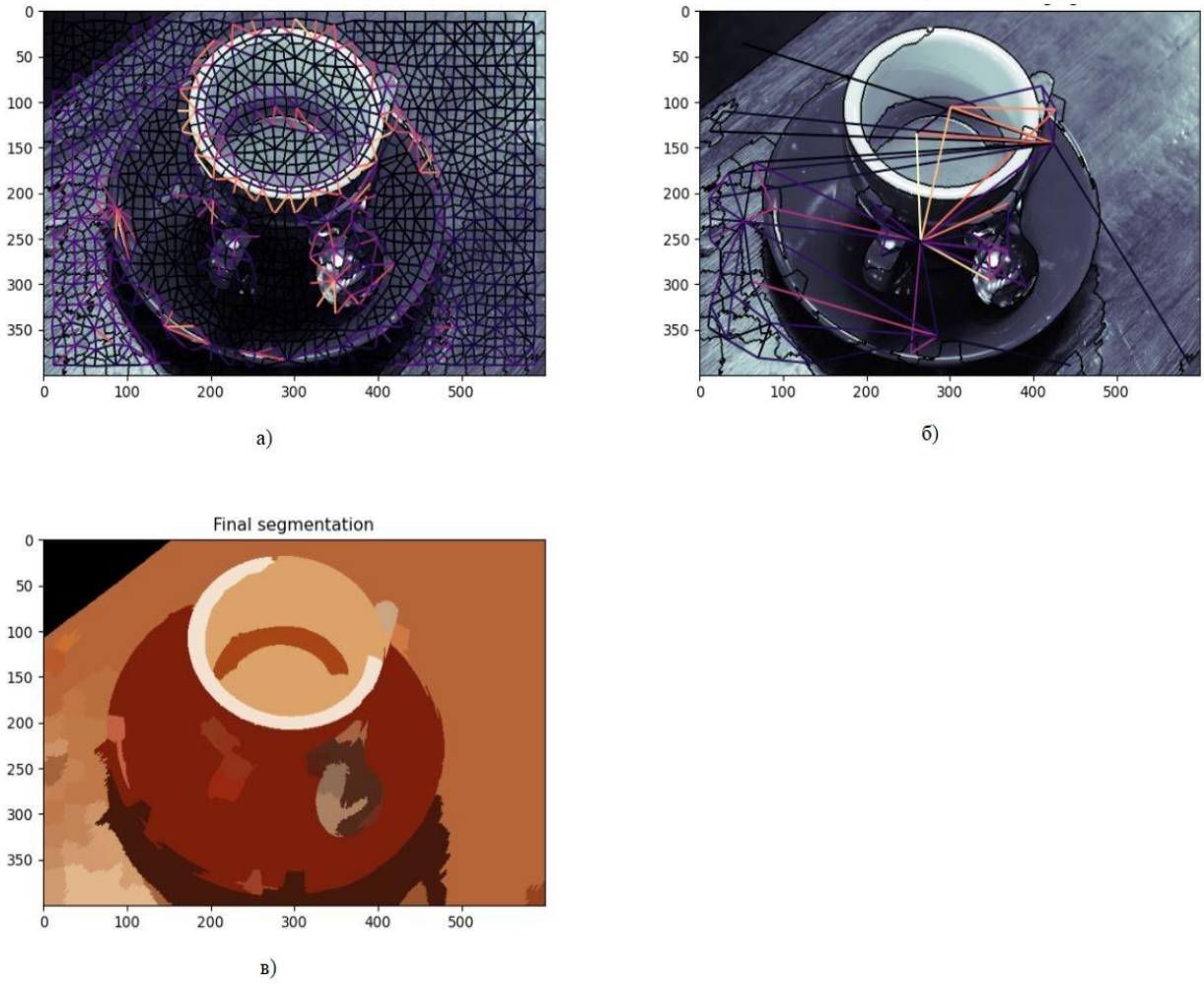


Рисунок 3.10 – Виконується ієрархічне складання: а) первинне розбиття; б) RAG після ієрархічного злиття; в) фінальна сегментація

3.6 Порівняння алгоритмів

Порівняємо алгоритми за часом, що в умовах ієрархічної сегментації є досить важливим та якістю об’єднання суперпікселів та коректного виявлення границь.

3.6.1 Порівняння на швидкодію

Для порівняння на швидкодію візьмемо декілька зображень із різними сценами, розміром та співвідношенням сторін та занесемо до таблиці 3.1 дані про роботу кожної ітерації.

Таблиця 3.1 – Дані про час виконання

Дослідження №	Алгоритм					
	FH Тест 1	SLIC Тест 1	FH Тест 2	SLIC Тест 2	FH Тест 3	SLIC Тест 3
1	36,65	11,22	11,29	11,94	16,32	9,29
2	0,27	0,5	0,26	0,49	0,22	0,53
3	0,25	0,58	0,29	0,43	0,22	0,39
4	0,26	1,26	0,27	0,65	0,23	0,82
5	0,25	13,82	0,27	13,79	0,23	11,28

Для більшої наочності побудуємо графік (рис. 3.11), на основі отриманих даних з таблиці 3.1 попередньо вирахувавши середнє значення для кожної ітерації кожного алгоритму.

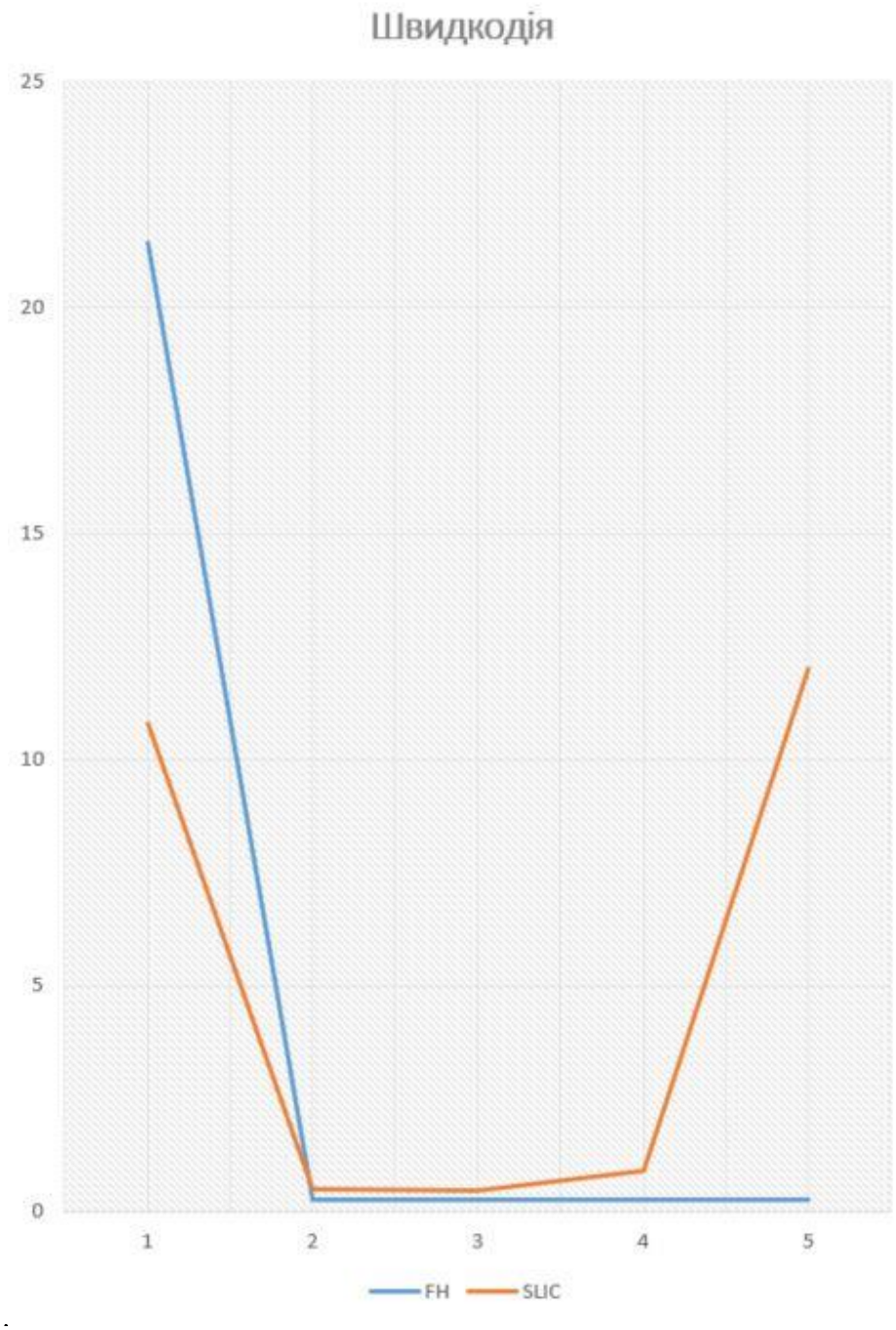


Рисунок 3.11 – Графік порівняння на швидкодію

З графіку видно, що перше розбиття зображення на суперпікселі алгоритму SLIC вдається набагато швидше за FH. Однак, слід зазначити, що алгоритми FH розбив зображення на більшу кількість суперпікселів. Далі можна спостерігати приблизно однаковий час на подальших ітераціях, однак чим менше залишається кластерів для об’єднання тім швидше починає працювати алгоритм FH.

3.6.2 Порівняння за якістю роботи з границям об’єктів

Для дослідження візьмемо два можливих сценарії використання суперпіксельної сегментації та розглянемо результати роботи досліджуваних алгоритмів.

3.6.2.1 Робота алгоритмів на супутниковому знімку

Перший дослід проведемо на основі супутникового знімку міста як зображено на рисунку 3.12.

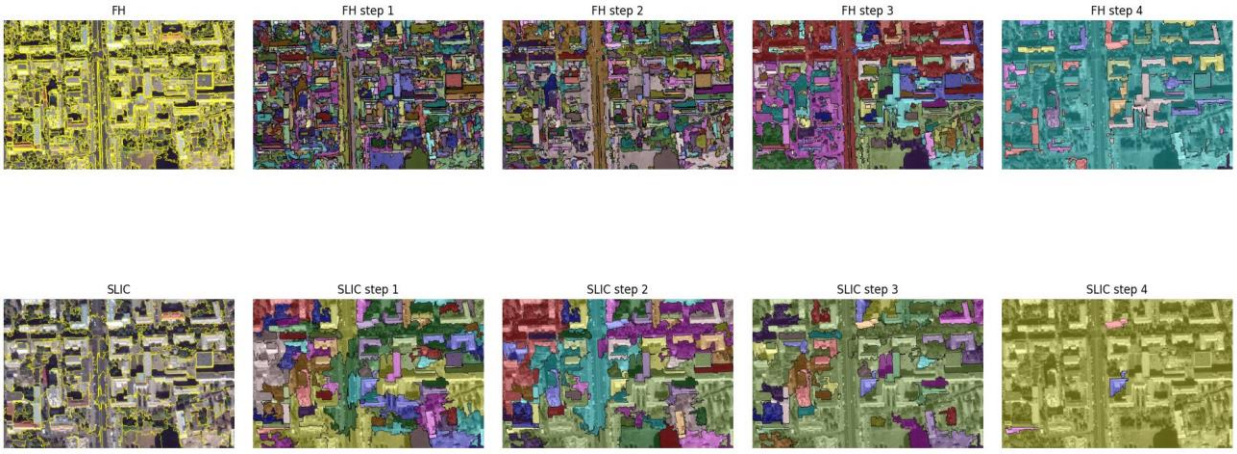


Рисунок 3.12 – Результат роботи

Розглянемо більш детально два останніх кроки роботи досліджуваних алгоритмів як зображено на рисунку 3.13.

Як видно, на останньому кроці алгоритм FH виділив більшу частину будівель, але також присутнє виділення більш незначних (малих) об’єктів як, наприклад, автівки та дерева. У той же час алгоритм SLIC максимально злив (об’єднав) суперпікселі.

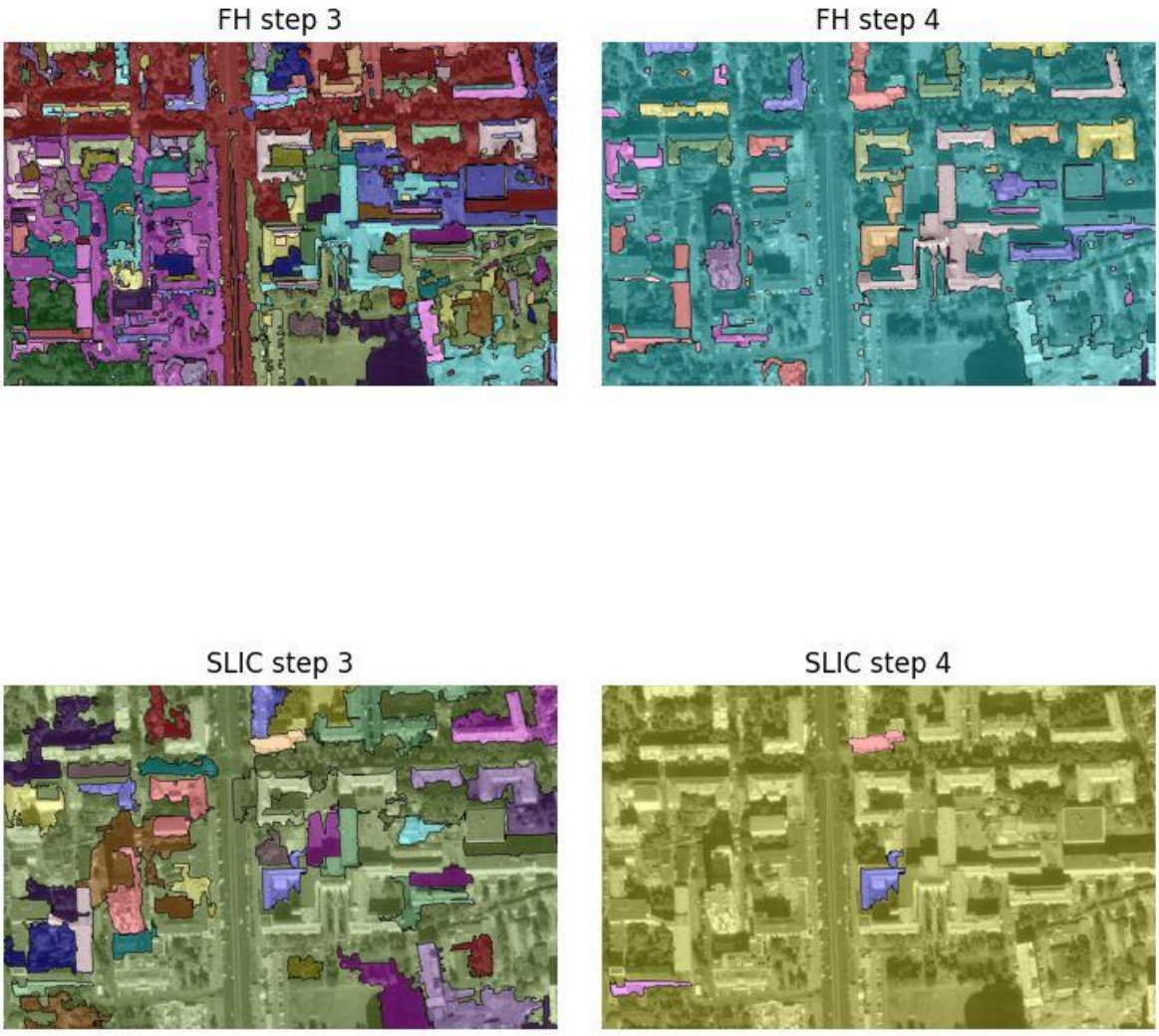


Рисунок 3.13 – Результат роботи

У даному виконанні коректніше буде порівнювати крок 4 для алгоритму FH та крок 3 для алгоритму SLIC. У такому випадку можна сказати, що кожен з алгоритмів має свої недоліки та переваги. У випадку із FH видно, що алгоритм виділи контури будівель та малі об’єкти, а алгоритм SLIC крім будівель виділив ще й зелені зони, у деяких випадках зливши їх із будівлями.

3.6.2.2 Робота алгоритмів на фото вулиці

Другий дослід проведемо на фото з відеорегістратору, моделюючи тим самим роботу алгоритмів у повсякденних задачах.

Результат роботи показано на рисунку 3.14.

Рисунок 3.14 – Результат роботи

Збільшимо область декілька важливих областей для більшої наочності роботи алгоритмів як зображено на рисунку 3.15, 3.16.

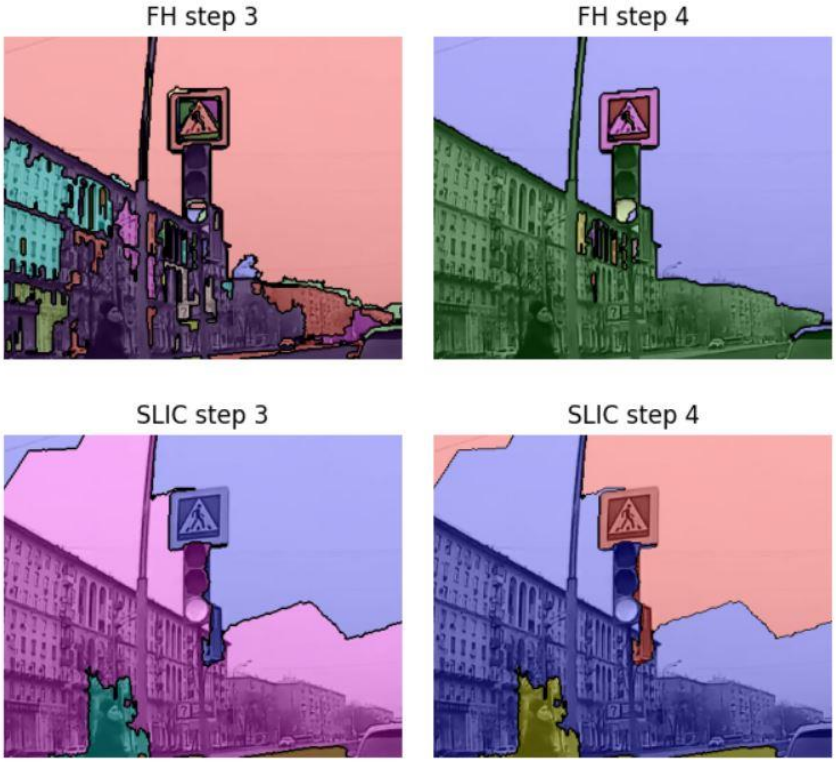


Рисунок 3.15 – Результат роботи

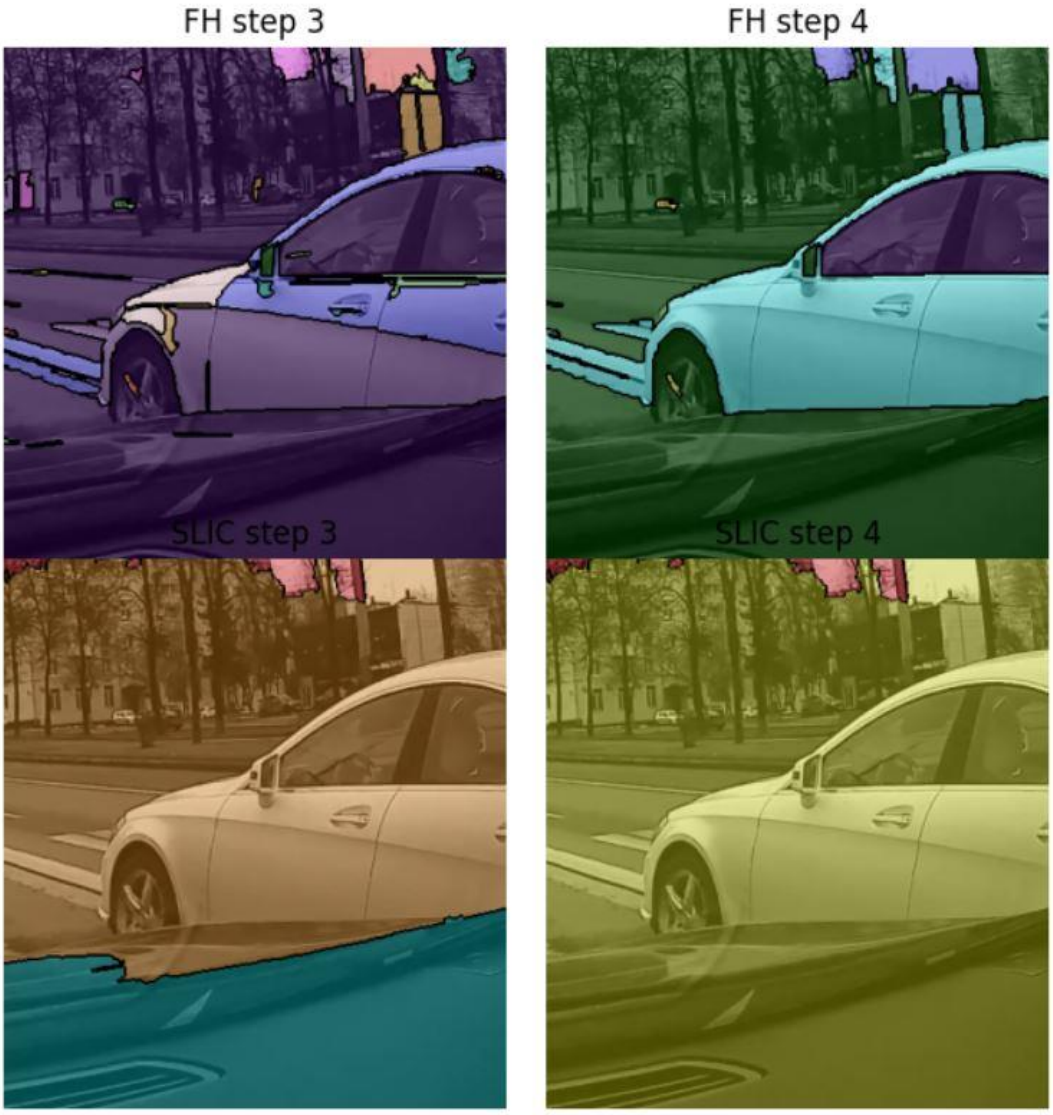


Рисунок 3.16 – Результат роботи

Як і під час попереднього досліду, на останньому кроці алгоритм FH має розбиття на більшу кількість суперпікселів, тому коректніше буде порівнювати крок 4 для алгоритму FH та крок 3 для алгоритму SLIC. У даному дослідженні досить гарно видно як алгоритм FH виділив границі будівлі, чітко відділивши її від неба, виділений дорожній знак та сигнал світлофора, лінії пішохідного переходу та автівку що проїжджає повз. У той же час алгоритму SLIC не вдалось коректно, відносно границь об’єктів на зображенні, об’єднати суперпікселі. При чому така тенденція прослідкується на усіх кроках. Не залежно від того на скільки укрупнені суперпікселі (на скільки суперпікселів розбито зображення.

ВИСНОВКИ

У рамках кваліфікаційної роботи було розроблено методи, що базуються на використанні алгоритмів суперпіксельної сегментації.

Використано методи суперпіксельної сегментації зображень, таких як: FH, SLIC.

Проведено дослідження методів ієрархічної суперпіксельної сегментації зображень, порівняння основних підходів до побудови алгоритмів сегментації, виміряно відносні показники швидкості та точності для різних алгоритмів. Отримані експериментальні дані свідчать про те, що найкращі результати досягаються при використанні сегментації методом FH.

Мету роботи досягнуто.

Результати дослідження апробовано у вигляді тез доповідей під час Міжнародної наукової конференції «CURRENT TRENDS IN THE DEVELOPMENT OF MODER SCIENTIFIC THOUGHT» [37].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Mashtalir, V. P., & Yakovlev, S. V. (2001). Point-set methods of clusterization of standard information. *Cybernetics and Systems Analysis*, *37*(3), 295-307.

2. Kinoshenko, D., Mashtalir, V., & Shlyakhov, V. (2007). A partition metric for clustering features analysis.

3. Mashtalir, V., Mikhnova, E., Shlyakhov, V., & Yegorova, E. (2006, November). A novel metric on partitions for image segmentation. In *2006 IEEE International Conference on Video and Signal Based Surveillance* (pp. 18-18). IEEE.

4. Kinoshenko, D., Mashtalir, V., & Yegorova, E. (2006). CLUSTERING METHOD FOR FAST CONTENTBASED IMAGE RETRIEVAL. In *Computer Vision and Graphics* (pp. 946-952). Springer, Dordrecht.

5. Bogucharskiy, S., & Mashtalir, V. (2015, September). Image segmentation via X-means under overlapping classes. In *2015 Xth International Scientific and Technical Conference" Computer Sciences and Information Technologies"(CSIT)* (pp. 45-47). IEEE.

6. Mashtalir, S., Mashtalir, V., & Stolbovyi, M. (2018, August). Representative based clustering of long multivariate sequences with different lengths. In *2018 IEEE second international conference on Data Stream Mining & Processing (DSMP)* (pp. 545-548). IEEE.

7. Богучарский, С. И., & Машталир, В. П. (2014). Сегментация изображений в больших базах данных с использованием плотности распределения информации. *Електротехнічні та комп'ютерні системи*, (14), 119-123.

8. Mashtalir, V., & Bogucharskiy, S. (2015). Covering Image Segmentation via Matrix X-means and J-means Clustering. *Sensors & Transducers*, *195*(12), 56.

9. Mashtalir, S., & Mashtalir, V. (2020). Spatio-temporal video segmentation. In *Advances in Spatio-Temporal Segmentation of Visual Data* (pp. 161-210). Springer, Cham.

10. Mashtalir, S., Mashtalir, V., & Stolbovyi, M. (2017). Video shot boundary detection via sequential clustering. *International Journal "Information Theories and Applications*, *24*(1), 50-59.

11. Mashtalir, S., & Mashtalir, V. (2016, August). Sequential temporal video segmentation via spatial image partitions. In *2016 IEEE First International Conference on Data Stream Mining & Processing (DSMP)* (pp. 239-242). IEEE.

12. Машталир, В. П., Шляхов, В. В., & Яковлев, С. В. (2014). Групповые структуры на фактор-множествах в задачах классификации. *Кибернетика и системный анализ*, (50,№ 4), 27-41.

13. Chupikov, A., Kinoshenko, D., Mashtalir, V., & Shcherbinin, K. (2006, July). Image retrieval with segmentation-based query. In *International Workshop on Adaptive Multimedia Retrieval* (pp. 207-221). Springer, Berlin, Heidelberg.

14. Bobbia, S., Macwan, R., Benezeth, Y., Nakamura, K., Gomez, R., & Dubois, J. (2021). Iterative Boundaries implicit Identification for superpixels Segmentation: a real-time approach. *IEEE Access*, *9*, 77250-77263.

15. Benesova, W., & Kottman, M. (2014). Fast superpixel segmentation using morphological processing. In *Conference on Machine Vision and Machine Learning* (pp. 67-1).

16. Jampani, V., Sun, D., Liu, M. Y., Yang, M. H., & Kautz, J. (2018). Superpixel sampling networks. In *Proceedings of the European Conference on Computer Vision (ECCV)* (pp. 352-368).

17. Reddy, V., Vaghdevi, K., & Kolli, D. (2021). DIGITAL IMAGE FORGERY DETECTION USING SUPER PIXEL SEGMENTATION AND HYBRID FEATURE POINT MAPPING. *European Journal of Molecular & Clinical Medicine*, *8*(2), 1485-1500.

18. Hamarneh, G., & Li, X. (2009). Watershed segmentation using prior shape and appearance knowledge. *Image and Vision Computing*, 27(1-2), 59-68.

19. Philipp-Foliguet, S., & Guigues, L. (2008). Multi-scale criteria for the evaluation of image segmentation algorithms. *Journal of multimedia*, 3(5), 42-56.

20. Galvão, F. L., Guimarães, S. J. F., & Falcão, A. X. (2020). Image segmentation using dense and sparse hierarchies of superpixels. *Pattern Recognition*, 108, 107532.

21. Di, S., Liao, M., Zhao, Y., Li, Y., & Zeng, Y. (2021). Image superpixel segmentation based on hierarchical multi-level LI-SLIC. *Optics & Laser Technology*, 135, 106703.

22. Xu, H., Langer, M., & Peyrin, F. (2021). Quantitative analysis of bone microvasculature in mouse model using the monogenic signal phase asymmetry and marker-controlled watershed. *Physics in Medicine & Biology*.

23. Howse, J., & Minichino, J. (2020). *Learning OpenCV 4 Computer Vision with Python 3: Get to grips with tools, techniques, and algorithms for computer vision and machine learning*. Packt Publishing Ltd.

24. Howse, J. (2013). *OpenCV computer vision with python*. Birmingham: Packt Publishing.

25. Shao, W., Sun, W., & Yang, G. (2021). Comparison of Texture Feature Extraction Methods for Hyperspectral Imagery Classification. *Remote Sensing Technology and Application*, *36*(2), 431-440.

26. Скорюкова, Я. Г., & Хмарук, С. М. (2021, March). КРИТЕРІЇ ОЦІНКИ МЕТОДІВ СЕГМЕНТАЦІЇ. In *L Науково-технічна конференція факультету будівництва, теплоенергетики та газопостачання (2021)*.

27. Mihaila, A. F., Penaru, P. S., Vlăsceanu, G. V., & Prodan, M. (2020). ON IMAGE SEGMENTATION USING A COMBINATION OF FELZENSZWALB, SLIC AND WATERSHED METHODS. *Journal of Information Systems & Operations Management*, 121-129.

28. Puri, S., & Singh, S. (2021). Image Segmentation Techniques. *International Journal of Engineering and Applied Physics*, *1*(2), 127-135.

29. Кошелев, Е. В. (2022). Исследование алгоритмов суперпиксельной сегментации для обработки в различных цветовых пространствах. *In Актуальные проблемы прикладной математики, информатики и механики* (pp. 1619-1624).

30. Tvoroshenko I., and Tkachenko D. (2020) Mechanisms of image classification based on descriptors of local features, Abstracts of IV International Scientific and Practical Conference «Integration of scientific bases into practice» (October 12-16, 2020). Stockholm, Sweden, pp. 443–448.

31. Wei, X., Yang, Q., Gong, Y., Ahuja, N., & Yang, M. H. (2018). Superpixel hierarchy. *IEEE Transactions on Image Processing*, 27(10), 4838-4849.

32. Vupputuri, A., Ashwal, S., Tsao, B., & Ghosh, N. (2020). Ischemic stroke segmentation in multi-sequence MRI by symmetry determined superpixel based hierarchical clustering. *Computers in Biology and Medicine*, 116, 103536.

33. Гороховатский В.А. (2003) Распознавание изображений в условиях неполной информации. Харків: ХНУРЭ, 112 с.

34. Tvoroshenko I., and Dziubenko M. (2020) Modern methods of analysis of the movement scheme using video detection of vehicles, Abstracts of V International Scientific and Practical Conference «Study of modern problems of civilization» (October 19-23, 2020). Oslo, Norway, pp. 422–428.

35. Gorokhovatskyi V., Putyatin Y., Gorokhovatskyi O., and Peredrii O. (2018) Quantization of the Space of Structural Image Features as a Way to Increase Recognition Performance, The Second IEEE International Conference on DataStream Mining & Processing 21-25 August 2018. Lviv, Ukraine, pp. 464–467.

36. Гороховатський В. О., Творошенко І. С., Чмутов Ю. В. (2022) Застосування систем ортогональних функцій для формування простору ознак у методах класифікації зображень, *Сучасні інформаційні системи,* 6(3), С. 5-12.

37. Філатов В. (2022) Аналіз алгоритмів суперпіксельної сегментації зображень, *Abstracts of I International Scientific and Practical Conference «Current trends in the development of modern scientific thought» (September 27 – 30, 2022). Haifa, Israel,* pp. 512-515.