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 ТА РЕАЛІЗАЦІЯ МЕТОДУ СУПЕРПІКСЕЛЬНОЇ СЕГМЕНТАЦІЇ ЗОБРАЖЕНЬ
(тема)

Виконав:
студент 2 курсу, групи ІНФм-21-1
Усік Д. П.
(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 Усіку Дмитру Павловичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження та реалізація методу суперпіксельної сегментації зображень

затверджена наказом по університету від 9 листопада 2022 року №1469Ст

2. Термін подання студентом роботи до екзаменаційної комісії 28 листопада 2022 р.

3. Вихідні дані до роботи науково-методична та науково-технічна література, математичні моделі 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 зображень.
2. Розробка застосунку для реалізації математичних моделей суперпіксельної сегментації.
3. Порівняння та аналіз алгоритмів сегментації зображень.

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 зображень.

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	доц. 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	11.11.22-13.11.22	
4	Аналіз проблеми	13.11.22-14.11.22	
5	Дослідження методів тестування	14.11.22-19.11.22	
6	Аналіз методів	19.11.22-25.11.22	
7	Оформлення пояснювальної записки	25.11.22-30.11.22	
8	Перевірка на плагіат	01.12.22	
9	Рецензування	02.12.21	
10	Підготовка презентації та доповіді	03.11.22-05.11.22	
11	Занесення роботи в електронний архів	06.12.22	
12	Попередній захист кваліфікаційної роботи	07.12.22	

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: 63 с., 2 табл., 25 рис., 1 дод., 45 джерел.

СУПЕРПІКСЕЛЬНА СЕГМЕНТАЦІЯ, АЛГОРИТМ СЕГМЕНТАЦІЇ SLIC, АЛГОРИТМИ ГЕНЕРАЦІЇ СУПЕРПІКСЕЛІВ.

Об’єктом дослідження є моделі суперпіксельної сегментаціїзображень об’єктів.

Метою дослідження є покращення методів суперпіксельної сегментації, які дозволяють детектувати ознаки (кольори).

Використано методи сегментації різних типів. Проведено дослідження методу сегментації SLIC. Досліджено метод сегментації SLIC з різними вхідними параметрами

У результаті роботи здійснена програмна реалізація алгоритму для сегментації данних.

SUPERPIXEL SEGMENTATION, SLIC SEGMENTATION ALGORITHM, SUPERPIXEL GENERATION ALGORITHMS.

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

The aim of the research is to improve the methods of superpixel segmentation, which allow to detect features (colors).

Different types of segmentation methods are used. A study of the SLIC segmentation method was performed. The SLIC segmentation method with different input parameters is investigated.

As a result, the software implementation of the algorithm for data segmentation.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

SLIC – Simple Linear Iterative Clustering (проста лінійна ітеративна кластеризація)

GDI – Graphics Device Inteerface (графічний інтерфейс користувача)

ООП – об’єктно орієнтоване програмування

NC – Normalized Cuts (нормовані зрізи)

GS – Graph-Based Image Segmentation (сегментація зображень на основі графів)

QS – Quick Shift (швидкий зсув)

TP – Turbopixel Algorithm (турбопіксельний алгоритм)

ВСТУП

Сегментація зображень, часто відома як перцептивне групування, є важливою сферою досліджень в обробці зображень. Вона відноситься до акту поділу зображення на численні сприймаються порівнянними частинами на основі їх спільних кольорів або текстур, або тому, що вони відносяться до речей, що представляють інтерес всередині зображення. Це питання широко досліджувалося протягом багатьох років і залишається досить активним і сьогодні.

Результати сегментації корисні для декількох застосувань комп'ютерного зору. У питаннях стерео і оцінки руху сегменти часто дають опорні області для обчислення відповідності. Методи сегментації традиційно використовуються для вирішення проблем розділення фігури і землі або розпізнавання по частинах в задачах більш високого рівня, таких як розпізнавання або індексування зображень.

Важливо підкреслити, що сегментація зображень і класифікація зображень є окремими завданнями. Для останньої бажаним результатом є позначення кожного пікселя класом, до якого він належить. Бачення сегментації зображення ширше і намагається виявити області, утворені пікселями зі схожими властивостями, і розбити всю картинку на ділянки, що відповідають різним компонентам сцени. При розгляді проблем наскрізної семантичної сегментації, в яких пікселі класифікуються для отримання сегментації, різниця стає нечіткою.

Сегментація також відрізняється від виявлення контурів, яке спрямоване на ідентифікацію контурів зображення. Існування різноманітних об'єктів позначається контурами, однак не всі контури є релевантними, залежно від розміру об'єкта, що цікавить, або поставленої візуальної задачі. Як правило,

виявлення контурів є початковим етапом сегментації. Однак, справжня сегментація досягається тільки після того, як відповідні контури були обрані і згруповані алгоритмом в замкнуті контури.

Сегментація зображення є складним процесом, і в даний час не існує повної теорії з цього питання, не в останню чергу тому, що сегментація часто призначена для конкретного використання. Алгоритми, засновані на градієнті, можуть бути порушені шумом або текстурованими патернами на зображенні, створюючи таким чином помилковий контур. Крім того, важко визначити місцезнаходження контуру, коли дві ділянки мають порівнянні кольори або рівні градацій сірого, або коли контур відповідає незначному градієнту. Крім того, результат сегментації залежить від застосування та розміру об’єктів, що представляють інтерес. Навіть сегментація, що виконується людиною, характеризується високим ступенем варіабельності результатів. Тому для деяких програм обробки зображень корисною є ієрархічна сегментація, яка забезпечує набір сегментів, пристосованих до різних масштабів.

Актуальність дослідження полягає у поліпшенні роботи алгоритмів суперпіксельної сегментації зображень, що демонструється на прикладі SLIC алгоритму.

1 ОГЛЯД МЕТОДІВ СЕГМЕНТАЦІЇ ЗОБРАЖЕНЬ

1.1 Суперпіксельна сегментація

Суперпіксельні алгоритми – це категорія підходів, призначених для поділу зображення на крихітні ділянки, які групують перцептивно пов’язані пікселі. Суперпіксельна сегментація часто використовується як етап попередньої обробки для задач комп’ютерного зору, таких як сегментація зображень, виявлення об’єктів, відстеження, оцінка глибини та класифікація об’єктів, щоб спростити обчислення для подальшої обробки або для забезпечення обчислювальної підтримки функцій [1].

Метою сегментації зображення є ідентифікація кожного пікселя відповідно до класу ймовірних об’єктів. Локалізація об’єктів і меж зображення допомагає знайти відповідне представлення для подальшого дослідження. Існує два основних методи сегментації зображень. Перший метод виділяє області, які відповідають заданому набору потенційних об’єктів [2]. Другий метод спрямований на виділення частин неідентифікованих класів об’єктів шляхом групування пікселів зі схожим кольором, текстурою і так далі, як показано на рисунку 1.1.

Перший метод максимізує ймовірність того, що кожен піксель позначений правильно. Використання класифікатора з ковзаючим вікном, який вгадує мітку кожного пікселя на основі його безпосереднього оточення, є найпростішим підходом. Більш складні алгоритми використовують когерентність (сусідні пікселі, ймовірно, мають схожі мітки) [1, 2] та семантичний контекст [3]. Вони часто використовують імовірнісні графічні моделі, такі як CRF [4]. У цьому випадку їх структура представлена графом,

вершинами якого є пікселі зображення, а ребрами – взаємозалежності між пікселями. Коли ребра допомагають присвоювати вершинам співставні мітки, стає досяжною ефективна сегментація.

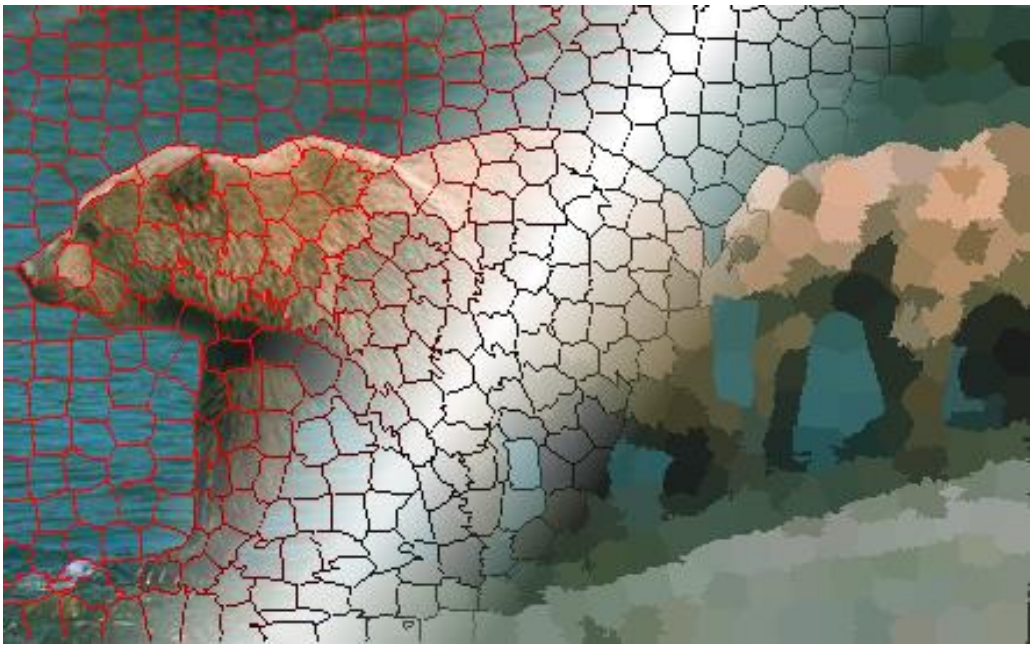


Рисунок 1.1 – Приклад роботи суперпіксельної сегментації

Метою другого методу є виділення найбільш ймовірних візуальних об’єктів з використанням якомога меншої кількості сегментів. Для вирішення подібних завдань набули популярності суперпіксельні алгоритми. Вони розбивають зображення на велику кількість відносно дрібних частин. Суперпіксельна сегментація допомагає зменшити розмірність зображення з невеликою втратою інформації і полегшує використання довгострокових взаємозалежностей між пікселями. Суперпікселі є корисним інструментом для визначення локальних дескрипторів зображення завдяки їх регулярності та здатності прилягати до кордонів зображення.

Загальними передумовами суперпіксельних алгоритмів є:

– продуктивність. Зазвичай суперпіксельна сегментація слугує технікою попередньої обробки. Відповідно, вона повинна займати менше часу, ніж необхідно для подальшої обробки;

– узгодженість. Межі суперпікселів повинні узгоджуватися з межами об’єктів зображення;

– компактність і регулярність. Для більшості застосувань суперпікселі повинні бути порівнянного розміру і приблизно опуклої форми. Такі суперпікселі сприяють вилученню більш точних локальних дескрипторів і мають менше сусідів.

1.2 Fast Marching Based Superpixels

Суперпіксельні сегментації – це суперпікселі, які дотримуються певних характеристик, таких як:

– дотримання меж: суперпікселі повинні підтримувати межі зображення. Ця характеристика має вирішальне значення, наприклад, для додатків сегментації. У літературі визначено декілька метрик для кількісної оцінки дотримання меж, включаючи відкликання меж, яке характеризує кількість граничних пікселів, відновлених алгоритмом сегментації з допуском 1 або 2 пікселі, та похибку недосегментації, яка кількісно оцінює «витік» суперпікселів, що перетинають фактичну межу зображення;

– рівномірність і чіткість: форма суперпікселів повинна бути правильною, а їхні краї – гладкими в областях, де немає межі об’єкта. Очевидно, що методи, які генерують суперпікселі з дуже звивистими контурами, з більшою ймовірністю демонструють високу прихильність до кордонів, але отриманий в результаті поділ рідко буває достатнім. Традиційно

форма суперпікселів характеризується такими критеріями, як середня компактність і щільність контуру суперпікселів;

– ефективність: суперпікселі повинні бути швидкими в обчисленні та ефективно використовувати пам’ять. Оскільки сегментація суперпікселів часто використовується як етап попередньої обробки для спрощення подальшої обробки, її застосування не повинно вимагати багато часу, особливо для операцій у реальному часі [5].

Метод Fast Marching Based Superpixels (FMS) заснований на подібності між хвилями, що поширюються в неоднорідному середовищі, і областями, що формуються на зображенні зі швидкістю, яка визначається локальним кольором і текстурою (рис. 1.2). Стаціонарне рівняння Ейконала описує поширення хвилі. Для того, щоб побудувати суперпіксельне розбиття, необхідно розв’язати рівняння Ейконала на області зображення для поля швидкостей, яке залежить від локального кольору і текстури. Рівняння для локальної швидкості як функції вмісту зображення, використання характеристик текстури та оновлення області під час розповсюдження відрізняються між двома методами.

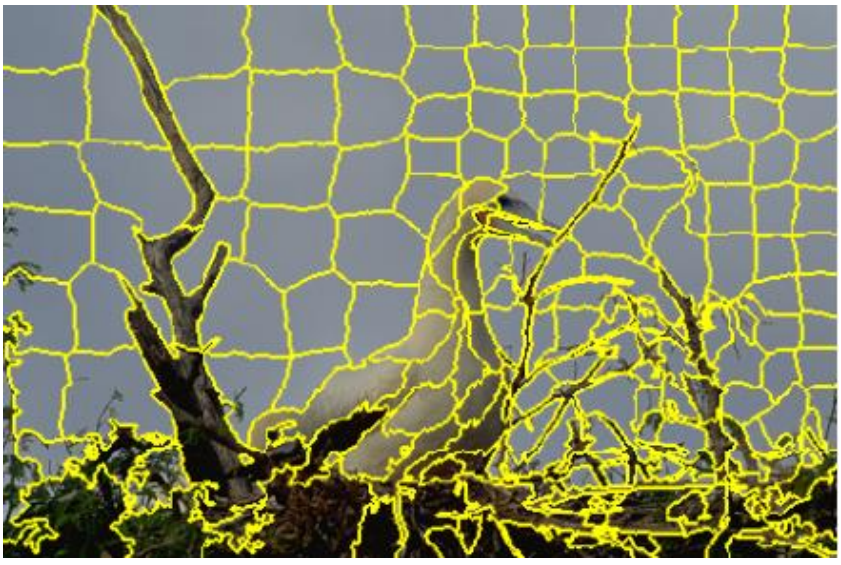


Рисунок 1.2 – Приклад виконання роботи алгоритму FMS

1.3 Ієрархічна сегментація на основі вейвлет-розкладання

Як було описано вище, сегментація зображень є класичною проблемою в обробці зображень, яка намагається визначити поділ зображення, в якому кожна виявлена область відповідає об’єкту, присутньому в сцені. Метод watershed transform є потужним математичним морфологічним інструментом для досягнення цієї конкретної мети [6]. Однак, при безпосередньому застосуванні до градієнта зображення, що підлягає сегментації, він, як правило, створює пересегментоване зображення. Для подолання цієї проблеми часто використовують маркери, які приблизно корелюють з положеннями об’єктів, що підлягають поділу. Тому найскладнішим аспектом сегментації, керованої маркерами, є ідентифікація позицій маркерів.

Watershed transform є відомим методом, заснованим на математичній морфології, який добре виконує завдання сегментації [7]. Його можна легко зрозуміти, провівши паралелі між зображенням і топографічним рельєфом. У цьому прикладі значення градації сірого певного пікселя розуміється як висота в певній позиції. Топографія рельєфу потім затоплюється водою, що виходить з найнижчої точки рельєфу. Коли вода з декількох мінімумів зустрічається на певній ділянці, така ділянка ідентифікується як край зображення. Як правило, алгоритм watershed transform використовується для сегментації градієнта зображення. Таким чином, кожен мінімум градієнта породжує окрему зону сегментації. Численні змінні, включаючи шум, помилку квантування та внутрішні текстури зображень, зазвичай призводять до того, що градієнтні оператори генерують значну кількість мінімумів. Як наслідок, загальновідомою проблемою методу watershed transform є те, що результуюче зображення, як правило, є надзвичайно пересегментованим.

Для вирішення проблеми надмірної сегментації одним з варіантів є застосування градієнтного оператора до попередньо відфільтрованих зображень.

Використання маркерів для виконання сегментації watershed transform є варіантом вирішення проблеми надмірної сегментації. Цей метод ґрунтується на припущенні, що положення об’єктів, які підлягають сегментації, можна наближено визначити. Планується проводити сегментацію за допомогою вейвлет-фільтрів, а не за допомогою градієнтних матриць [8].

1.4 Дослідження подібностей між регіонами

У різноманітних методах сегментації зображень початковим кроком є обчислення суперпіксельного поділу зображення. Суперпікселі значно зменшують візуальне представлення. Як правило, наступні фази процедури сегментації намагаються згрупувати суперпікселі разом, щоб створити сегментоване зображення.

Кластеризація суперпікселів вимагає двох важливих компонентів. По-перше, має бути визначено точне визначення подібності (або відмінності) між сусідніми суперпікселями. По-друге, має бути визначена відповідна техніка кластеризації для того, щоб виконати об’єднання суперпікселів з використанням вищезгаданої відстані.

Метою сегментації зображення є розбиття зображення I на сімейство областей або сегментів $\{S_0, S_1, ..., S_N\}$, де N – бажана кількість сегментів, а S_i – група пікселів всередині одного сегмента: $\{S_i = p : l(p) = i\}$, де кожен сегмент позначається i [9].

Алгоритми машинного навчання часто використовуються для класифікації або регресії і можуть бути умовно розділені на три групи: контрольовані, неконтрольовані та напівконтрольовані.

1.5 Об’єднання регіонів

Зображення графів суміжності областей є важливим компонентом численних методів сегментації, таких як алгоритм нормалізованого розрізу та алгоритм водоспаду.

Ми можемо описати зображення як граф, використовуючи його пікселі або набір пікселів, таких як суперпікселі або області, утворені технікою суперсегментації, як вузли, з’єднуючи сусідні пікселі або області ребрами, і вибираючи міру подібності або відмінності як вагу ребра. Перевагою сегментації на основі графів є те, що вона враховує просторову інформацію, оскільки сусідні області або пікселі пов’язані між собою.

Використання графових представлень для сегментації зображень сягає корінням у найдавніші часи обробки зображень. Перше покоління підходів використовує мінімальне остовне дерево (MST) графа суміжності областей.

Мінімальне остовне дерево графа, також відоме як найкоротше остовне дерево (SST), охоплює вершини графа з найменшою вагою. Остовне дерево зв’язного неорієнтованого графа – це підмножина ребер, що з’єднує всі вершини без циклів і з найменшою можливою сумарною вагою ребер. Існують ефективні методи обчислення MST графа, наприклад, алгоритм Крускала [10].

1.6 Постановка задачі дослідження

За умовами поставленого завдання необхідно дослідити метод, що розпізнає та сегментує зображення. Проєктування та розробка повинна бути зроблена відповідно до поставлених вимог та строків.

Таким чином, можна зробити висновок, що сегментація є актуальним завданням обробки зображень та розпізнавання образів. З цією метою поставлено завдання розробки алгоритмів суперпіксельної обробки, тобто сегментації зображення на групи однорідних, непересічних пов’язаних областей (група пікселів, також звані суперпікселями).

Об’єктом дослідження є моделі суперпіксельної сегментаціїзображень об’єктів.

Метою дослідження є покращення методів суперпіксельної сегментації, які дозволяють детектувати ознаки (кольори).

Для досягнення мети необхідно вирішити такі завдання:

– провести аналіз існуючих методів сегментування зображень;

– розробити алгоритм сегменцації суперпікселів;

– реалізувати комп’ютерну модель для суперпіксельного сегментування зображень.

2 МАТЕМАТИЧНІ МОДЕЛІ АЛГОРИТМІВ СУПЕРПІКСЕЛЬНОЇ СЕГМЕНТАЦІЇ ЗОБРАЖЕНЬ

2.1 Суперпіксельні алгоритми

Методи отримання суперпікселів можна умовно класифікувати як засновані на графах або на градієнтному підйомі. Нижче ми розглянемо основні підходи до отримання суперпікселів для кожної з цих категорій, включаючи деякі з них, які спочатку не призначалися для отримання суперпікселів. А також проведемо їх якісний та кількісний аналіз і порівняємо відносну продуктивність.

2.1.1 Алгоритми на основі графів

Графові методи генерації суперпікселів розглядають кожен піксель як вершину графа. Ваги ребер між парою вузлів пропорційні схожості пікселів. Створення суперпікселів передбачає мінімізацію функції вартості, заданої на графі.

NC05 – Алгоритм Normalized cuts розділяє мережу всіх пікселів зображення ітеративно, використовуючи контурні та текстурні підказки, мінімізуючи функцію вартості, визначену на ребрах на границях розбиття. Він генерує дуже однорідні, естетично красиві суперпікселі, як можна побачити на рисунку 2.1. Однак, NC05 має досить погане дотримання меж і є найповільнішим підходом (особливо для великих фотографій), незважаючи на спроби прискорити алгоритм.

Рисунок 2.1 – Приклад роботи алгоритму NC05

GS04 – Felzenszwalb та Huttenlocher пропонують альтернативний метод на основі графів для створення суперпікселів. Він кластеризує пікселі як вузли мережі так, що кожен суперпіксель є найнижчим остовним деревом його складових пікселів [11]. Фактично, GS04 ефективно дотримується меж зображення, але генерує суперпікселі з дуже випадковими розмірами та формами (рис. 2.2). Цей алгоритм покладається на локальні градієнти зображення з метою рівномірного розподілу суперпікселів на площині зображення.

SL08 – Мур пропонує представляють метод генерації суперпікселів, які прилягають до сітки шляхом знаходження оптимальних шляхів, або швів, розділяючих зображення на менші вертикальні та горизонтальні ділянки. За допомогою техніки, схожої на Seam Carving, визначаються оптимальні шляхи.

Рисунок 2.2 – Приклад роботи алгоритму GS04

Векслер у своїй роботі застосовує стратегію глобальної оптимізації для обох GCa10 (рис. 2.3) та GCb10 (рис. 2.4) алгоритмів, подібну до тієї, що використовується у їхній роботі з синтезу текстур [12]. Суперпікселі створюються шляхом з’єднання фрагментів зображення, що перекриваються, таким чином, щоб кожен піксель належав лише до однієї з ділянок, що перекриваються. Вони пропонують два варіанти своєї технології, один для створення компактних суперпікселів (GCa10), а інший для створення суперпікселів постійної інтенсивності (GCb10).Так, щоб кожен піксель відповідав лише одній з ділянок зображення зони перекриття. Вони пропонують дві варіації свого методу: один для отримання компактних суперпікселів (GCa10) і один для отримання суперпікселів, що мають постійну інтенсивність (GCb10).

Рисунок 2.3 – Приклад роботи алгоритму GCa10

Рисунок 2.4 – Приклад роботи алгоритму GCb10

2.1.2 Алгоритми на основі градієнтного сходження

Методи градієнтного сходження починаються з грубого початкового групування пікселів і ітеративно модифікують кластери до тих пір, поки не буде виконана умова збіжності для генерації суперпікселів.

MS02 – Зсув середнього значення, ітераційний метод пошуку моди для виявлення локальних максимумів функції густини, використовується для виявлення закономірностей у просторі характеристик кольору або інтенсивності зображення. Суперпікселі визначаються пікселями, які сходяться до однієї моди [13]. MS02 – це більш рання методика, яка генерує суперпікселі дивної форми та розміру.

QS08 – Швидкий зсув також використовує стратегію сегментації, що шукає режим. Сегментація ініціюється за допомогою методу медоїдного зсуву. Потім кожна точка в просторі ознак переміщується до свого найближчого сусіда, таким чином збільшуючи оцінку щільності Парзена. Розмір та кількість суперпікселів не можуть бути явно контрольовані в QS08. QS08 використовувався в попередніх дослідженнях для локалізації об’єктів та сегментації руху.

WS91 – Метод watershed використовує градієнтний підйом, що починається з локальних мінімумів, для генерації вододілів, які є лініями, що розділяють водозбірні басейни.

Отримані в результаті суперпікселі часто мають дуже нерівномірний розмір і форму, з поганим зчепленням кордонів [14].

TP09 – Turbopixel метод поступово розширює сукупність місць розташування насіння, використовуючи геометричний потік на основі заданого рівня.

Геометричний потік залежить від локальних градієнтів зображення для рівномірного розподілу суперпікселів по площині зображення [15]. На відміну від WS91, суперпікселі TP09 повинні бути однаковими за розміром, щільністю та зчепленням з межами. TP09 залежить від алгоритмів різної складності, але фактично він є одним з найповільніших оцінюваних алгоритмів і демонструє погану граничну адгезію.

2.2 Алгоритми суперпіксельної генерації

SLIC алгоритм приймає на вхід задану кількість K суперпікселів приблизно однакового розміру. Оціночний розмір кожного суперпікселя на зображенні з N пікселів становить, відповідно, N/K пікселів. Для суперпікселів майже однакового розміру в кожному інтервалі сітки $S = \sqrt{N/K}$ буде центр суперпікселя.

На початку цього процесу вибирається K центрів кластерів суперпікселів $C_k = [l_k, a_k, b_k, x_k, y_k]^T$ на регулярних інтервалах сітки S, де $k = [1, K]$. Оскільки просторова протяжність будь-якого суперпікселя становить близько S^2 (приблизний розмір суперпікселя), можна зробити висновок, що пікселі, пов’язані з цим центром кластера, знаходяться в межах області $2S \times 2S$ навколо центру суперпікселя в площині xy. Це стає областю пошуку пікселів, найближчих до центру кожного кластера.

Малі евклідові відстані в колірному просторі CIELAB мають перцептивне значення (m в рівнянні 1). Якщо просторові відстані між пікселями перевищують цю межу, вони починають переважувати схожість інтенсивностей пікселів [16] (що призводить до появи суперпікселів, які не враховують межі областей, а лише близькість у площині зображення). Замість

того, щоб використовувати стандартну евклідову норму в 5D просторі, ми використовуємо міру відстані, яка позначається наступним чином:

$$d_{lab} = \sqrt{(l_k - l_i)^2 + (a_k - a_i)^2 + (b_k - b_i)^2}, \tag{2.1}$$

$$d_{xy} = \sqrt{(x_k - x_i)^2 + (y_k - y_i)^2}, \tag{2.2}$$

$$D_s = d_{lab} + \frac{m}{S} d_{xy}, \tag{2.3}$$

де *Ds* – сума *lab* відстані та відстані по площині *xy*, нормованої на інтервал сітки *S*.

В *Ds* введена змінна *m*, яка дозволяє контролювати компактність суперпікселя. Чим більше значення *m*, тим більше підкреслюється просторова близькість і тим компактніше кластер. Ця величина може знаходитись в діапазоні [1, 20]. Для всіх результатів у цій роботі ми обираємо $m = 10$. Це приблизно відповідає емпіричному максимуму перцептивно значущої відстані CIELAB і забезпечує хороший баланс між схожістю кольорів та просторовою близькістю [17].

Робота алгоритму починається з вибірки *K* центрів кластерів з регулярними інтервалами та перенесення їх у початкові місця, що відповідають місцю з найнижчим градієнтом в околі 3 × 3. Це робиться для того, щоб запобігти розташуванню їх на краю і обмежити ймовірність вибору пікселя з шумом. Градієнти зображення розраховуються як:

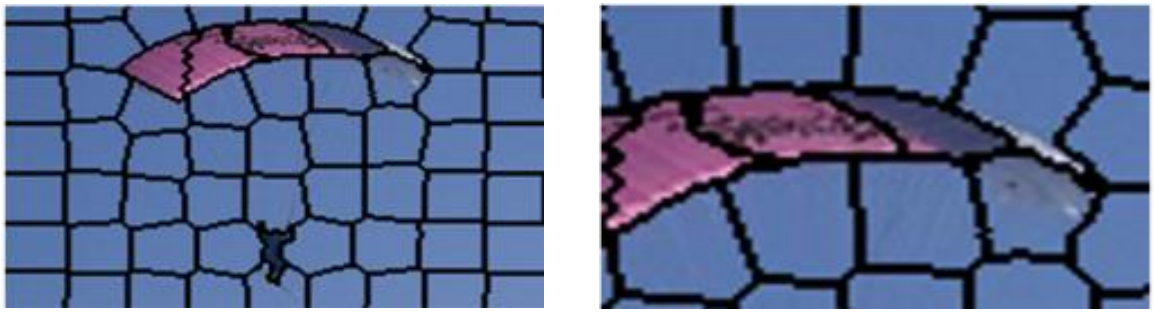
$$G(x,y) = \| I(x+1,y) - I(x-1,y) \|^2 + \\ + \| I(x,y+1) - I(x,y-1) \|^2, \quad (2.4)$$

де *I(x, y) – lab* вектор, що відповідає пікселю в позиції *(x, y)*, та ||.|| – норма L_2. При цьому враховується інформація як про колір, так і про інтенсивність.

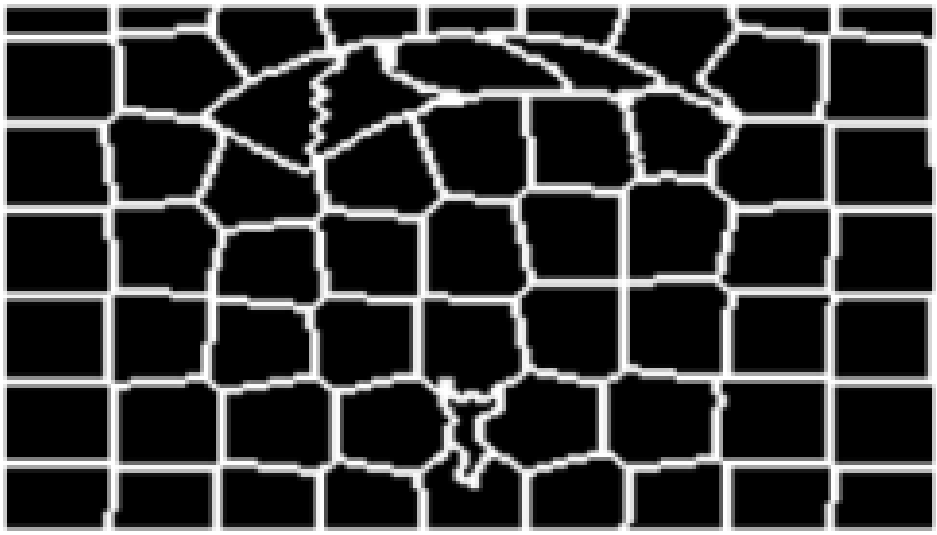
Кожному пікселю на зображенні відповідає найближчий центр кластера, область пошуку якого перекриває цей піксель. Після співставлення кожного пікселя з найближчим центром кластера генерується новий центр як середнє значення вектора *labxy* всіх пікселів кластера. Процедура зв'язування пікселів з найближчим центром кластера та перерахунку центру кластера повторюється до тих пір, поки не буде досягнута збіжність [18].

На завершення цієї операції може існувати декілька блукаючих міток, тобто декілька пікселів в області більшого сегмента з такою ж міткою, але не пов'язаних з ним. Незважаючи на метрику просторової близькості, це можливо, оскільки наша кластеризація не гарантує зв'язність в явному вигляді. На останньому етапі нашого підходу ми забезпечуємо зв'язок шляхом перепозначення роз'єднаних сегментів мітками найбільшого найближчого кластера. Ця операція вимагає менше 10% від загального часу, необхідного для сегментації зображення (рис. 2.5).

Алгоритм SEEDS базується на ідеї максимізації енергії і продовжує розвиток підходів до суперпіксельної сегментації зображень. Цільова енергетична функція *E* розбиття на суперпікселі *s* розраховується як сума двох величин.



а) б)



в)

Рисунок 2.5 – Приклад роботи алгоритму SLIC:
а) первинне розбиття; б) ділянка зображення після розбиття;
в) сітка розбиття

Перша величина *H(s)* характеризує однорідність кольору в рамках суперпікселів, друга величина *G(s)* є опціональною та характеризує однорідність границь суперпікселів:

$$E(s) = H(s) + yG(s), \tag{2.5}$$

$$H(s) = \sum_{K} \psi(C_{A_K}), \tag{2.6}$$

$$C_{A_K}(j) = \frac{1}{Z} \sum_{i \in A_K} \delta(I(i) \in Hj), \tag{2.7}$$

$$\psi(C_{A_K}) = \sum_{\{H_j\}} (C_{A_K}(j))^2, \tag{2.8}$$

де $\psi(C_{A_K})$ – функція міри якості розподілення кольору в межах кластеру *k*;

C_{Ak} визначає гістограму кольору множини пікселів A_k, що утворюють кластер *k*;

$\delta(I(i) \in Hj)$ – індикаторна функція, що вертає 1 коли колір пікселю *I(i)* опиняється в камері гістограми *j*.

Для розрахунку величини *G(s)* навколо кожного пікселя і будується патч N_i. За анологією з величиноюрозподілення кольору, друга величина розраховуються на основі гістограми міток суперпікселів, що присутні в межах патчу:

$$b_{N_i}(k) = \frac{1}{Z} \sum_{j \in N_i} \delta(j \in A_K), \tag{2.9}$$

$$G(s) = \sum\nolimits_i \sum\nolimits_K (b_{N_i}(k))^2. \tag{2.10}$$

2.3 Automatic Quick-Shift Segmentation method

Підхід має на меті вилучення суперпікселів шляхом автоматичного налаштування параметрів швидкого зсуву на основі виявлення та розпізнавання об’єктів – корисних інваріантних зображень [19].

По-перше, застосовується інваріантна техніка, щоб виключити елементи, які можуть змінити захоплення зображення, такі як тінь і підсвічування. Для вектора-стовпчика 3D $Cs = [Cs(R), Cs(G), Cs(B)]^T$, що представляє реакції червоного, зеленого і синього датчиків системи кольорового зображення, яка бачить об’єкт, інваріантне рівняння має вигляд:

$$C_s^{`(i)} = \frac{C_s^{(i)} - \min\{C_s^{(R)}, C_s^{(G)}, C_s^{(B)}\}}{\sqrt{\sum_{j\in\{R,G,B\}} (C_s^{(i)} - \min\{C_s^{(R)}, C_s^{(G)}, C_s^{(B)}\})^2}}, \quad (2.11)$$

для $i=\{R,G,B\}$.

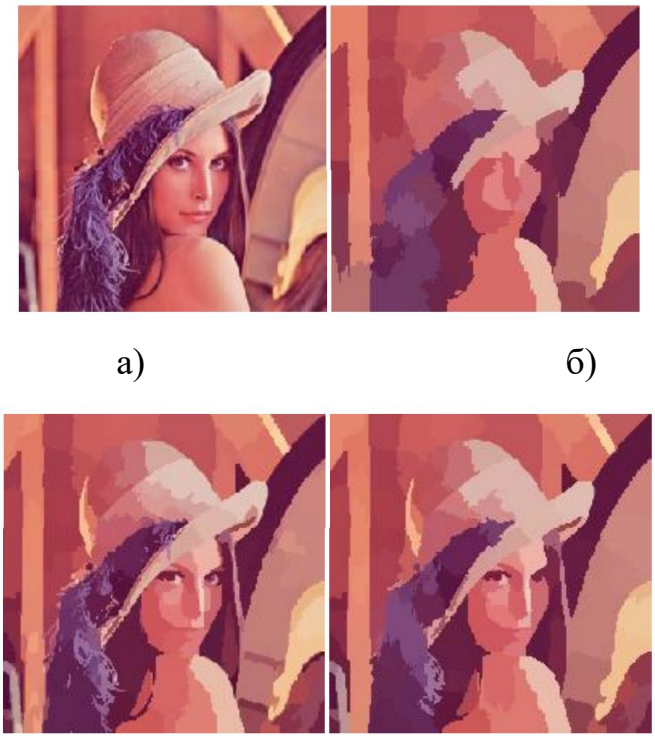
Таке зображення є доступним для всіх матеріальних поверхонь, включаючи діелектричні та металеві, якщо розглядати його в умовах загального освітлення кольоровими джерелами світла. Однак, інваріант потребує більше часу, якщо розмір зображення перевищує 250 × 250, тому для зменшення часу обробки інваріанту доводиться розбивати зображення на частини залежно від його розміру.

Далі використовується квантування, щоб мінімізувати кількість кольорів на інваріантному зображенні перед ручною сегментацією.

При вимірюванні схожості в якості еталонного зображення використовується зображення, сегментоване вручну.

Квантування – це процес зменшення кількості кольорів на зображенні шляхом розбиття колірного куба RGB на ряд менших блоків, а потім відображення всіх кольорів, які потрапляють всередину кожного блоку, до центрального значення цього блоку [20].

Виділення суперпікселів з інваріантного зображення для порівняння з сегментованим вручну зображенням з використанням методу швидкого зсуву. Нижче можна побачити, що метод є чутливим до вибору параметрів, тому ручне налаштування є недостатнім. На рисунку 2.6 порівнюється налаштування параметрів швидкого зсуву вручну та автоматично.



а) б)

в) г)

Рисунок 2.6 – Приклад порівняння налаштуваннь параметрів:

а) оригінальне зображення;

б) ручне налаштування з невеликою кількістю сегментів;

в) ручне налаштування з великою кількістю сегментів;

г) автоматичне налаштування

Як видно з рисунку 2.6 б), неправильний вибір числа для швидкого зсуву може призвести до втрати даних зображення або фрагментації зображення на надто малу кількість сегментів. Як видно з рисунку 2.6 в), це також може призвести до невиправданого перевищення кількості сегментів або поділу зображення на надто велику кількість частин. На рисунку 2.6 г) видно, що вибір значень параметрів швидкого зсуву робить сегментовані зображення більш релевантними і легшими для вивчення.

2.4 Лінійна спектральна суперпіксельні кластеризація

LSC (Linear Spectral Clustering Superpixel) – алгоритм, який не тільки генерує суперпікселі з найсучаснішим приляганням до границь, але й фіксує глобальні атрибути зображення. На основі дослідження зв’язку між цільовими функціями нормалізованих зрізів та зваженим *K*-середнім представлено метод LSC. Покращення цих двох цільових функцій є порівнянними, якщо подібність між цими двома точками у вхідному просторі дорівнює зваженій функції ядра між двома відповідними векторами у високорозмірному просторі ознак зі складним дизайном. Таким чином, відносно складний власний підхід до мінімізації нормованої цільової функції розкрою може бути замінений простим зваженим кластером *K*-середніх у цьому просторі ознак. На відміну від кластеризації зваженого ядра *K*-середніх [7, 21], LSC виключає побудову масивної матриці ядра, а вимога збіжності може бути виконана природним чином. LCS досягає лінійної складності шляхом поступового обмеження простору пошуку зважених *K*-середніх, зберігаючи при цьому відмінну якість отриманих суперпікселів.

$$F_{k-m} = \sum_{k=1}^{K} \sum_{p \in \pi_k} w(p) \| \phi(p) - m_k \|^2, \tag{2.12}$$

$$m_k = \frac{\sum_{q \in \pi_k} w(q)\phi(q)}{\sum_{q \in \pi_k} w(q)}, \tag{2.13}$$

$$F_{Ncuts} = \frac{1}{K} \sum_{K=1}^{K} \frac{\sum_{p \in \pi_k} \sum_{q \in \pi_k} W(p,q)}{\sum_{p \in \pi_k} \sum_{q \in V} W(p,q)}, \tag{2.14}$$

де m_k – центр k-го кластеру π_k;

p та q – пікселі з вагою $w(p)$ та $w(q)$ відповідно;

ϕ визначає функцію, що зв'язує пікселі з багатовимірним просторм ознак;

$W(p,q)$ – близкість між двома точками p та q;

V – множина всіх вершин графу.

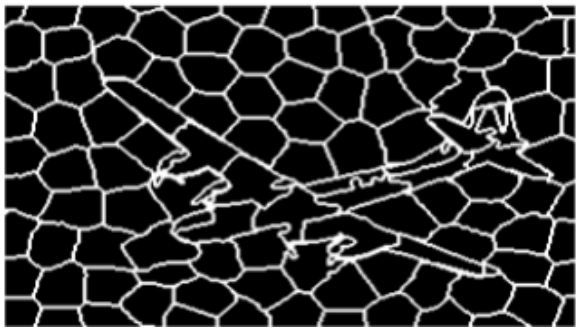
Для того, щоб можна було використовувати зважений метод k-середніх, що приблизно буде еквівалентний методу нормалізованих зрізів, формується відповідний десятивимірних простір ознак на основі просторового розташування та значення кольору у просторі CIELAB:

$$\phi(p) = \frac{1}{w(p)}(C_c \cos\frac{\pi}{2}1_p, \sin\frac{\pi}{2}1_p, 2.55C_c \cos\frac{\pi}{2}\alpha_p,$$
$$2.55C_c \sin\frac{\pi}{2}\alpha_p 2.55C_c \cos\frac{\pi}{2}\beta_p 2.55C_c \sin\frac{\pi}{2}\beta_p, \quad (2.15)$$
$$C_c \cos\frac{\pi}{2}x_p, C_c \sin\frac{\pi}{2}x_p, C_c \cos\frac{\pi}{2}y_p, C_c \sin\frac{\pi}{2}y_p,$$

$$w(p) = \sum_{q \in V} W(p,q) = w(p)\phi(p) \cdot \sum_{q \in V} w(q)\phi(p), \quad (2.16)$$

де C_c та C_s використовуються для контролю відносних коефіцієнтів колірної та просторової інформації відповідно [22]. Роботу алгоритму продемонстровано на рисунку 2.7.

а) б)



в)

Рисунок 2.7 – Приклад роботи алгоритму LSC:
а) первинне розбиття; б) ділянка зображення після розбиття;
в) сітка розбиття

2.5 Compact Wateshed

Алгоритм сегментації Compact Wateshed є досить ефективним. Однак він має нерівномірні за розмірами та сформовані сегменти, а також дуже мінливі межі. Користувач не має можливості контролювати характеристики сегментації. У цьому розділі пояснюється наша проста стратегія введення регульованого обмеження компактності в сегментацію водозбору для того, щоб оцінити його вплив на якість та характеристики сегментації [23]. У цій програмі висока компактність означає, що суперпікселі мають приблизно однаковий розмір і приблизно правильну форму (наприклад, прямокутник або коло) за відсутності сильних градієнтів зображення.

Інтуїтивно зрозуміла концепція походить з географії: коли ландшафт затоплюється падаючими краплями води, виникають басейни, заповнені водою, і хребти, що їх розділяють, в залежності від кількості води. Цими хребтами і є вододіли. Оскільки сегментація водозборів є добре відомим алгоритмом, існує декілька його реалізацій та модифікацій.

OpenCV підтримує сегментацію водозборів із затравками (також відому як сегментація водозборів, керованих маркерами). Метод отримує ділянки із зовнішнього джерела, такі як локальні мінімуми градієнта або суперпіксельна сегментація на рівномірній сітці. Пікселі зростають піксель за пікселем до тих пір, поки не досягнуть межі сегмента, що оточує інший піксель. Ці межі є вододілами. Функція відстані використовується для визначення наступного елемента, який повинен вирости на один піксель [24]. У реалізації OpenCV ця функція відстані просто враховує інтенсивність або значення кольору пікселя. Це призводить до надзвичайно мінливих меж в однорідних областях зображення і потенційно дуже неправильної форми сегментів, коли присутні градієнти зображення.

На рисунку 2.8 зображено значний вплив на візуальний вигляд, форму, розмір та розподіл сегмента на площині зображення.

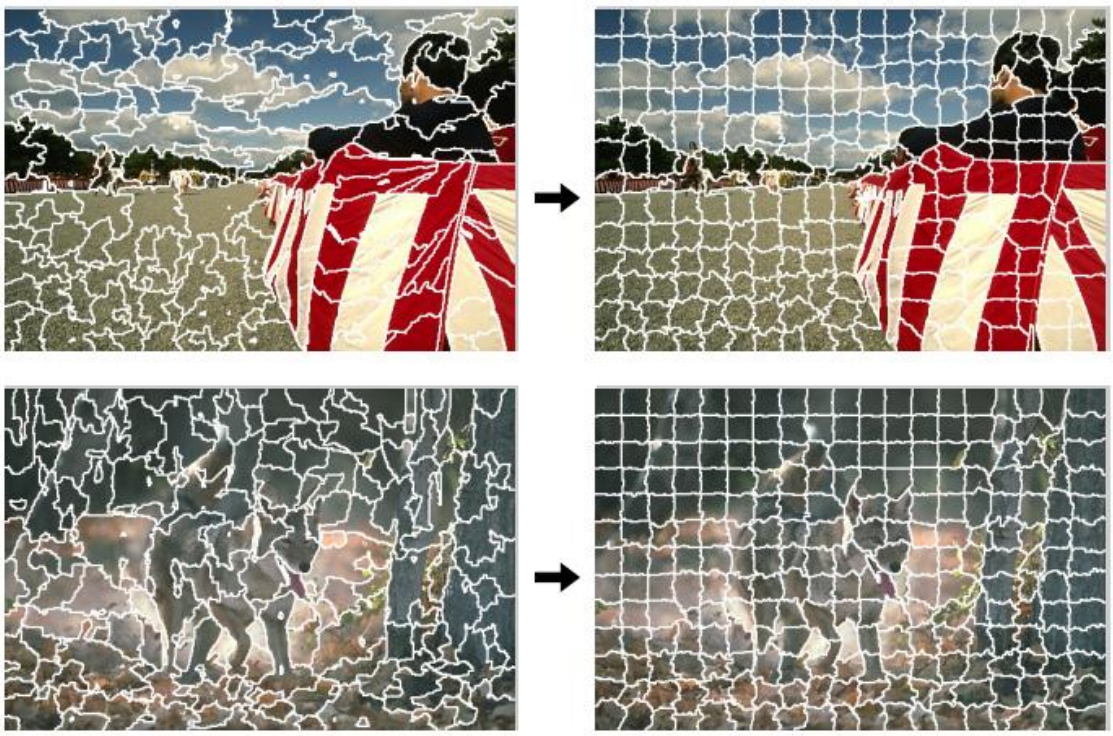


Рисунок 2.8 – Порівняння результатів роботи алгоритмів Watershed та Compact Watershed

2.6 Порівняння алгоритмів сегментації

2.6.1 Ефективність обчислень і використання пам’яті

Суперпікселі часто використовуються для заміни піксельної сітки з метою прискорення інших алгоритмів. Тому важливо, щоб суперпікселі можна було ефективно створювати з самого початку. На рисунку 2.9 досліджується кількість часу, необхідного декільком алгоритмам сегментації суперпікселів для сегментації зображень зростаючого розміру. SLIC, з його складністю *O(N)*, є найшвидшою суперпіксельною технікою, і його перевага зростає зі

збільшенням розміру зображення. Хоча складність GS04 $O(N)log\ N$ є конкурентоспроможною, решта алгоритмів демонструють значну різницю в продуктивності обробки [25].

Суперпіксельний метод також повинен бути ефективним щодо пам’яті, щоб обробляти величезні зображення. SLIC є найбільш ефективним методом, який використовує лише N значень з плаваючою комою для кодування відстані між кожним пікселем та його найближчим центром кластера. Інші методи використовують значно більше пам’яті: Для чотирьох з’єднань GS04 і GC10 вимагають $5N$ плаваючих значень для зберігання крайових ваг і порогів (або $9N$ для восьми з’єднань).

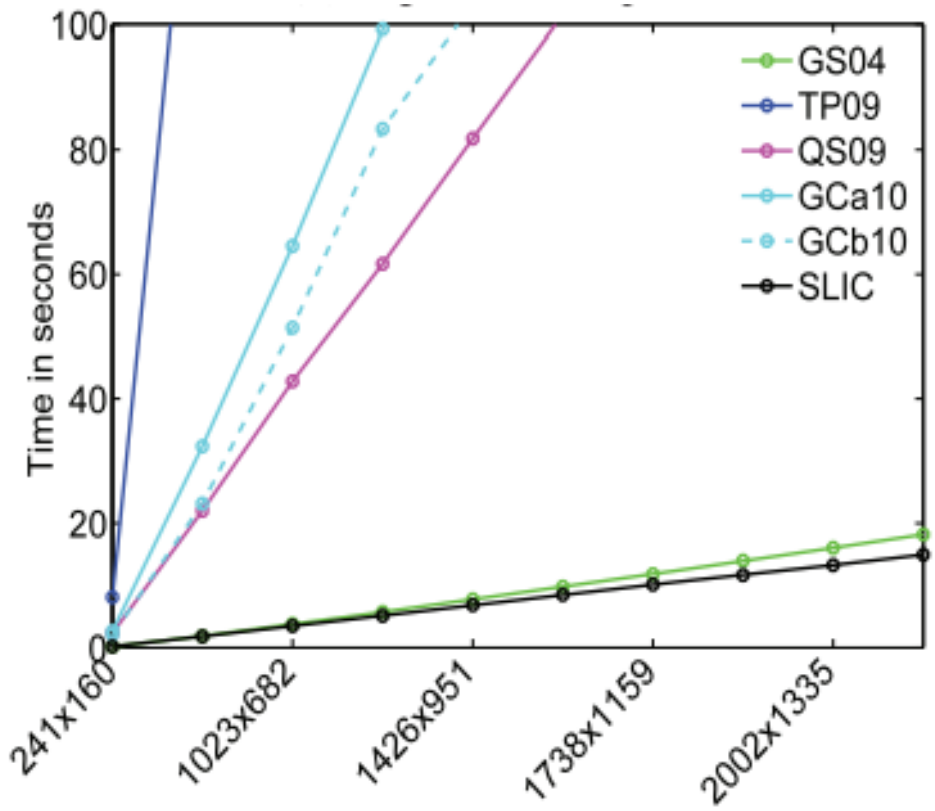


Рисунок 2.9 – Час необхідний алгоритмам для генерації суперпікселів на зображенні

2.6.2 Дотримання меж

Здатність дотримуватися меж зображення є, мабуть, найбільш важливою характеристикою суперпіксельної техніки. Стандартні вимірювання дотримання меж включають помилки пригадування меж та недосегментації; SLIC, GS04, NC05, TP09, QS09 та GC10 порівнюються на рисунках 2.10 та 2.11 з використанням цих метрик з бази даних Берклі. Крім того, «Квадрати» відносяться до базової продуктивності, досягнутої шляхом сегментації зображення на рівномірні квадрати [22, 26]. Набір даних Berkeley складається з 300 зображеннь розміром 321 × 481 піселів, кожне з яких має близько 10 сегментацій базової істини, що коментуються людиною.

Метрика відкликання границь підраховує частку базових ребер, які знаходяться всередині двох пікселів суперпіксельної границі. На рисунку 2.10 показано відображення границь кожного підходу в залежності від кількості суперпікселів. Сильний відгук границь означає, що мало фактичних границь було пропущено.

Найкращу здатність до запам’ятовування кордонів мають суперпікселі SLIC та GS04. Якщо зменшити компактність SLIC *m* зі значення за замовчуванням 10, то SLIC покаже кращі результати, ніж GS04 [23, 27].

На рисунку 2.11 зображено помилку недосегментації, яка є ще одним показником дотримання кордонів.

Зазвичай алгоритми сегментації використовують суперпікселі як етап попередньої обробки. Високоякісний метод суперпікселів має підвищити продуктивність методу сегментації, з яким він використовується. На наборі даних MSRC було порівняно сегментацію, отриману за допомогою SLIC, GS04, NC05, TP09, QS09 та GC10 [28].

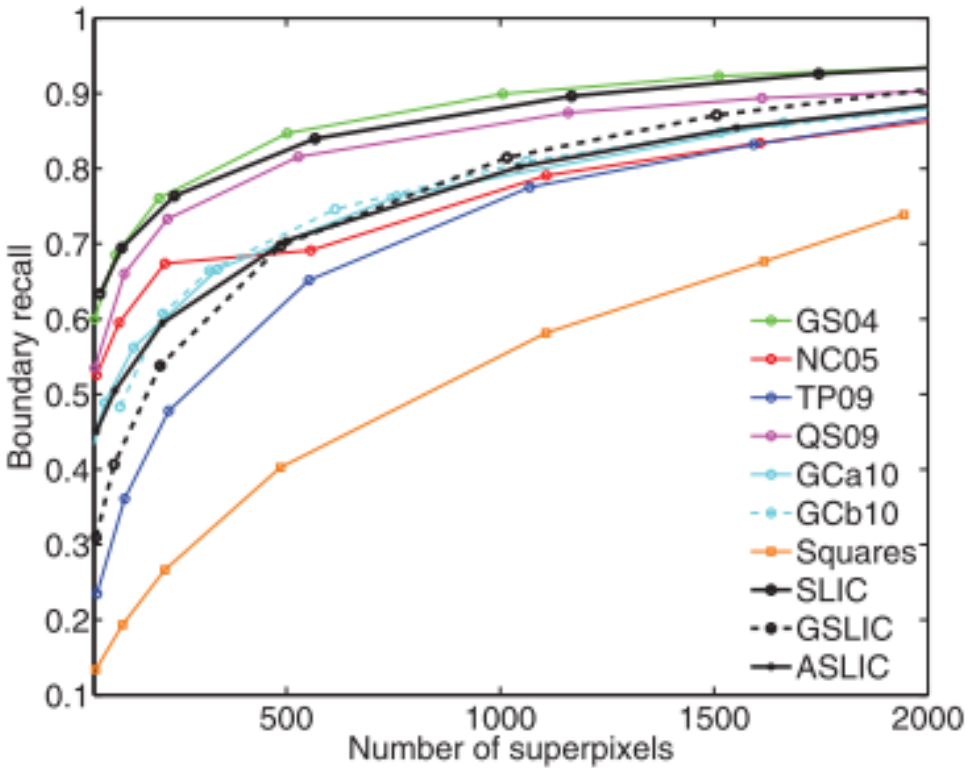


Рисунок 2.10 – Порушення меж сегментації зображення

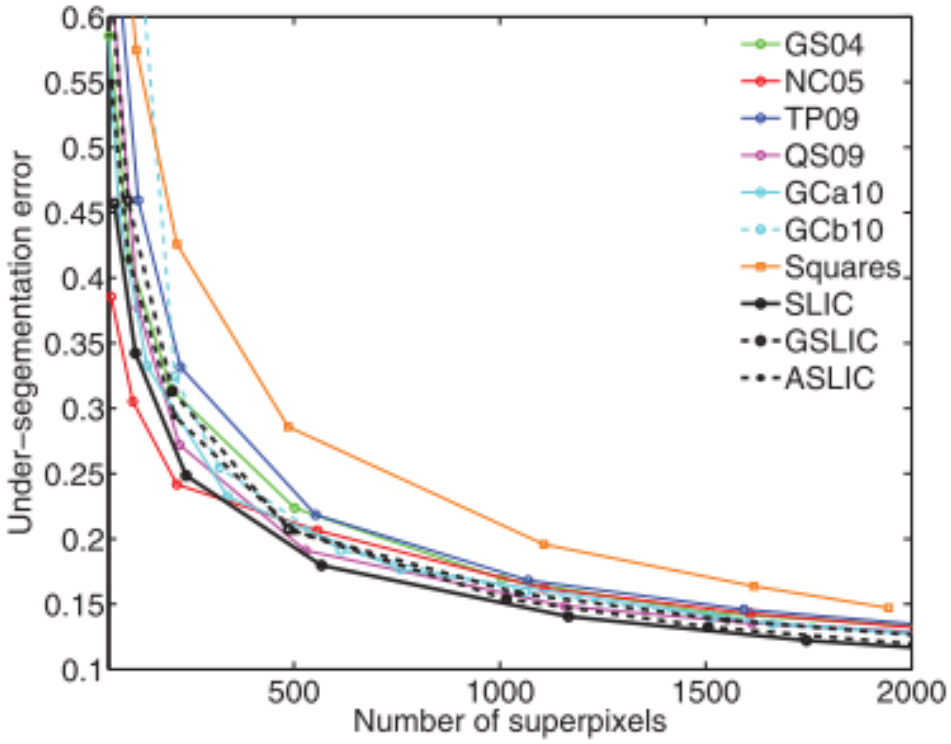


Рисунок 2.11 – Помилка недосегментації

Підхід, який використовує суперпікселі для обчислення даних про колір, текстуру, геометрію та місцезнаходження, дав такі результати. Потім розробляється CRF-модель та навчаються класифікатори для 21 класу об’єкту [29]. Таблиця 2.1 демонструє, що суперпікселі SLIC забезпечують найвищу продуктивність. Крім того, SLIC економить обчислювальний час приблизно в 500 разів порівняно з NC05 (табл. 2.2).

Таблиця 2.1 – Порівняння основних алгоритмів на основі графів

Критерій порівняння	Алгоритми на основі графів			
	GS	**NC**	**GCa**	**GCb**
1	**2**	**3**	**4**	**5**
Помилка недосегментації	0,23	0,22	0,22	0,22
Порушення меж сегментації зображення	0,84	0,68	0,69	0,70
Швидкість сегментації для зображення 320 × 240 пікселів	1,08 с	178,15 с	5,30 с	4,12 с
Швидкість сегментації для зображення 2048 × 1536 пікселів	90,95 с	-	315 с	235 с

Продовження таблиці 2.1

1	2	3	4	5
Точність сегментації	74,6%	75,9%	-	73,2%

Таблиця 2.2 – Порівняння основних алгоритмів на основі градієнтного сходження

Критерій порівняння	Алгоритми на основі градієнтного сходження		
	TP	**QS**	**SLIC**
Помилка недосегментації	0,24	0,20	0,19
Порушення меж сегментації зображення	0,61	0,79	0,82
Швидкість сегментації для зображення 320 × 240 пікселів	8,10 с	4,66 с	0,36 с
Швидкість сегментації для зображення 2048 × 1536 пікселів	800 с	181 с	14,94 с
Точність сегментації	62,0%	75,1%	76,9%

3 КОМП'ЮТЕРНА МОДЕЛЬ СУПЕРПІКСЕЛЬНОЇ СЕГМЕНТАЦІЇ ЗОБРАЖЕНЬ

3.1 Обґрунтування вибору середовища програмної реалізації

В рамках кваліфікаційної роботи був розроблений алгоритм обробки зображень на основі методу суперпіксельної сегментації. Для реалізації було обрано інтегроване середовище розробки для мови програмування Java – IntelliJ IDEA. Це пов'язано з тим, що наявні в IntelliJ IDEA засоби автозавершення та аналізу коду, швидкого виявлення помилок та швидких виправлень підвищують ефективність процесу розробки.

Граматика мови сформувалася під значним впливом мов C та C++. Зокрема, в її основі лежить об'єктна модель C++, проте вона була вдосконалена та модифікована. Процес проектування об'єктно–орієнтованих систем полегшується за рахунок виключення певних конфліктних сценаріїв, які можуть виникнути внаслідок помилки програміста. Різноманітні завдання, які в C/C++ є обов'язком програміста, делегуються віртуальній машині.

По–перше, оскільки мова Java розроблялася як платформонезалежна, вона має менше низькорівневих можливостей для взаємодії з апаратним забезпеченням, що впливає на продуктивність програми, на відміну, наприклад, від мови C++. Java дозволяє викликати підпрограми, написані іншими мовами програмування, коли це необхідно.

Під «архітектурною незалежністю» мається на увазі, що програмне забезпечення, написане на Java, буде працювати на будь-якій підтримуваній апаратній або системній платформі без модифікації вихідного коду або перекомпіляції.

Це може бути досягнуто шляхом компіляції вихідного коду Java в байт-код, який є більш простими машинними командами. Згодом додаток може бути виконаний на будь-якій платформі, яка містить віртуальну машину Java, яка переводить байт-код в код, специфічний для операційної системи та процесора. В даний час існують віртуальні машини Java для більшості процесорів і операційних систем.

Доступ до платформно-залежної функціональності, такої як обробка графіки, багатопотоковість та мережева робота, полегшується за допомогою стандартних бібліотек. У деяких версіях байт-код може бути перетворений в машинний код до або під час виконання програми для підвищення продуктивності JVM.

Переносимість є однією з основних переваг байт-коду. Java вважається повільнішою, ніж мови, які компілюються в машинний код, через додатковий час, необхідний для цього кроку через додаткові витрати на інтерпретацію.

Однією з таких технік є компіляція just-in-time (JIT), яка перетворює байт-код Java в машинний код і кешує його при першому виконанні програми. Як наслідок, така програма запускається і виконується швидше, ніж базовий інтерпретований код, але за рахунок додаткових витрат на компіляцію під час виконання. Частковим вирішенням цієї проблеми є використання віртуальних машин, які оцінюють поведінку програми та частково перекомпілюють її з метою оптимізації окремих ділянок коду [30]. Крім того, це може виявити так звані «гарячі точки», тобто ділянки програми, як правило, всередині циклів, які потребують найбільшого часу для виконання. Без шкоди для переносимості, JIT-компіляція та динамічна перекомпіляція підвищують продуктивність Java-програм.

Існує також альтернативний підхід до оптимізації байт-коду, відомий як статична компіляція або компіляція з випередженням часу (AOT). На додаток

до типових компіляторів, ця техніка використовує пряму компіляцію в машинний код. Це забезпечує вищу продуктивність, ніж інтерпретація, але в жертву переносимості, оскільки програма може бути виконана тільки на цільовій платформі.

Java, на відміну від C++, є більш об’єктно-орієнтованою. Об’єктні класи організовують всю інформацію та операції. Винятком з повної об’єктної орієнтації (як у Smalltalk) є примітивні типи (int, float тощо). Це було свідоме рішення, прийняте винахідниками мови для підвищення швидкості. Тому Java не вважається повністю об’єктно-орієнтованою мовою програмування [31].

В Java всі об’єкти є похідними від первинного об’єкта, який називається просто Об’єкт.

Хоча в C++ вперше було введено множинне успадкування, Java підтримує тільки одинарне успадкування, що запобігає потенційним конфліктам між членами класу (методами і змінними), успадкованими від базових класів.

Протягом життя об’єкту Java використовує автоматичний збирач сміття (GC – Garbage Collector) для управління пам’яттю. Програміст визначає, коли створювати об’єкти, а віртуальна машина відповідає за звільнення пам’яті, коли об’єкт більше не потрібен. Коли немає більше посилань на даний об’єкт, збирач сміття може автоматично видалити його з пам’яті. Тим не менш, витік пам’яті все одно може статися, якщо код програміста містить посилання на більш непотрібні об’єкти, наприклад, ті, що зберігаються в активних контейнерах.

Збірка сміття допускається в будь-який момент. В ідеалі це відбувається під час неактивного стану програми. Коли в купі не вистачає вільної пам’яті для обробки нового об’єкта, автоматично спрацьовує збірка сміття, що може викликати кількасекундне зависання. Отже, існують версії віртуальної

машини Java зі збирачем сміття, які розроблені спеціально для створення систем реального часу.

3.2 Бібліотека обробки зображень OpenCV

OpenCV – це сучасна бібліотека з відкритим вихідним кодом, яка використовується в комп'ютерному зорі, машинному навчанні та обробці зображень.

З її допомогою можна розпізнавати предмети або людські обличчя на фотографіях та відео. Java може обробляти структуру масивів OpenCV для аналізу в поєднанні з багатьма бібліотеками. Векторний простір та математичні операції над цими ознаками використовуються для ідентифікації візуального шаблону та його численних характеристик.

OpenCV – бібліотека алгоритмів комп'ютерного зору, обробки зображень та чисельних алгоритмів загального призначення з відкритим кодом. Реалізована C/C++, також розробляється для Python, Java, Ruby, Matlab, Lua та інших мов [32]. Може вільно використовуватися в академічних та комерційних цілях – поширюється за умов ліцензії BSD.

OpenCV 2 включає серйозні зміни в інтерфейсі C++, спрямовані на спрощення, поліпшення безпеки, введення нових функцій і збільшення продуктивності (особливо для багатоядерних систем). Офіційні релізи тепер випускаються кожні шість місяців.

Самі бібліотеки:

– Microsoft Windows: компілятори Microsoft Visual C ++ (6.0, .NET 2003),Intel Compiler, Borland C ++, Mingw (GCC 3.x);

– Windows RT: портований ARM компанією Itseez;

– Linux: GCC (2.9x, 3.x), Intel Compiler: «./configure–make–make install», RPM (spec файл включений у постачання);

– Mac OS X: GCC (3.x, 4.x);

– Android;

– iOS;

– використовуються C та «полегшений» C++. Прагми та умовна компіляція використовуються дуже обмежено.

Засоби GUI, захоплення відео:

– Microsoft Windows: DirectShow, VfW, MIL, CMU1394;

– Linux: V4L2, DC1394, FFMPEG;

– Mac OS X: QuickTime.

Основні модулі бібліотеки:

Cxcore – це ядро, що містить базові структури даних та алгоритми:

– базові операції над багатовимірними числовими масивами;

– матрична алгебра, математичні функції, генератори випадкових чисел;

– запис/відновлення структур даних у/з XML;

– базові функції 2D графіки CV;

– модуль обробки зображень та комп'ютерного зору;

– базові операції над зображеннями (фільтрація, геометричні перетворення, перетворення колірних просторів тощо);

– аналіз зображень (вибір відмітних ознак, морфологія, пошук контурів, гістограми);

– аналіз руху, стеження за об'єктами;

– виявлення об'єктів, зокрема осіб;

– калібрування камер, елементи відновлення просторової структури.

Highgui – модуль для введення/виводу зображень і відео, створення інтерфейсу користувача:

– захоплення відео з камер та відео файлів, читання/запис статичних зображень;

– функції для організації простого UI (всі демо-програми використовують HighGUI).

Cvaux – експериментальні та застарілі функції:

– просторів. зір: стерео калібрація, саме калібрація;

– пошук стерео-відповідності, кліки у графах;

– знаходження та опис рис особи.

CvCam – захоплення відео дозволяє здійснювати захоплення відео з цифрових відеокамер (підтримка припинена і в останніх версіях цей модуль відсутній)

У версії 2.2 бібліотека була реорганізована. Замість універсальних модулів cxcore, cvaux, highGUI та інших було створено кілька компактних модулів з більш вузькою спеціалізацією:

– opencv_core – основна функціональність. Включає базові структури, обчислення (математичні функції, генератори випадкових чисел) і лінійну алгебру, DFT, DCT, для XML і YAML і т. д.;

– opencv_imgproc – обробка зображень (фільтрація, геометричні перетворення, перетворення колірних просторів тощо);

– opencv_highgui – простий UI, введення/виведення зображень та відео;

– opencv_ml – моделі машинного навчання (SVM, дерева рішень, навчання зі стимулюванням тощо);

– opencv_features2d – розпізнавання та опис плоских примітивів (SURF(англ.) FAST та інші, включаючи спеціалізований фреймворк);

– opencv_video – аналіз руху та відстеження об’єктів (оптичний потік, шаблони руху, усунення фону);

– opencv_objdetect – виявлення об’єктів на зображенні (знаходження

осібза допомогою алгоритму Віоли-Джонса, розпізнавання людей HOG і т. д.);

– opencv_calib3d – калібрування камери, пошук стерео-відповідності та елементи обробки тривимірних даних;

– opencv_flann – бібліотека швидкого пошуку найближчих сусідів (FLANN 1.5) та обгортки OpenCV;

– opencv_contrib – супутній код, який ще не готовий для застосування;

– opencv_legacy – застарілий код, збережений для зворотної сумісності;

– opencv_gpu – прискорення деяких функцій OpenCV за рахунок CUDA, створеного за допомогою NVidia.

3.3 Програмна реалізація

Для програмної реалізації було обрано дослідження алгоритму SLIC, а в якості мови програмування – Java, з використанням бібліотеки OpenCV. Приклад написаної програми наведено на рисунку 3.1.

```
import java.io.*;
import java.util.*;

class SLIC {

    private int[] dx4 = {-1, 0, 1, 0};
    private int[] dy4 = {0, -1, 0, 1};
    private int m_width;
    private int m_height;
    private double[] m_lvec = null;
    private double[] m_avec = null;
    private double[] m_bvec = null;

void RGB2XYZ(int sR, int sG, int sB, double[] XYZ) {
        double R = sR/255.0;
        double G = sG/255.0;
        double B = sB/255.0;

        double r, g, b;
        if (R<=0.04045) {
            r=R/12.92;
        } else {
            r= Math.pow((R+0.055)/1.055, 2.4);
        }
        if (G<=0.04045) {
            g=G/12.92;
        } else {
            g=Math.pow((G+0.055)/1.055, 2.4);
        }
        if (B<=0.04045) {
            b=B/12.92;
        } else {
            b=Math.pow((B+0.055)/1.055, 2.4);
        }
        XYZ[0] = r*0.4124564 + g*0.3575761 + b*0.1804375;
        XYZ[1] = r*0.2126729 + g*0.7151522 + b*0.0721750;
        XYZ[2] = r*0.0193339 + g*0.1191920 + b*0.9503041;
    }
}
```

Рисунок 3.1 – Реалізація алгоритму SLIC

Алгоритм здатен розбивати зображення на різну кількість сегментів, в залежності від заданих параметрів (рис. 3.2).

а)

б)

Рисунок 3.2 – Робота програми:

а) сегментація на 400 сегментів; б) сегментація на 100 сегментів;

3.4 Управління застосунком

Перш за все необхідно інсталіювати мову програмування Java, на персональний комп’ютер, налаштувати середовище розробки, підключити

бібліотеку OpenCV для обробки зображень та Math для наукових досліджень та розрахувань.

Для цього потрібно з офіційної сторіки завантажити дистрибутив Java (рис. 3.3).

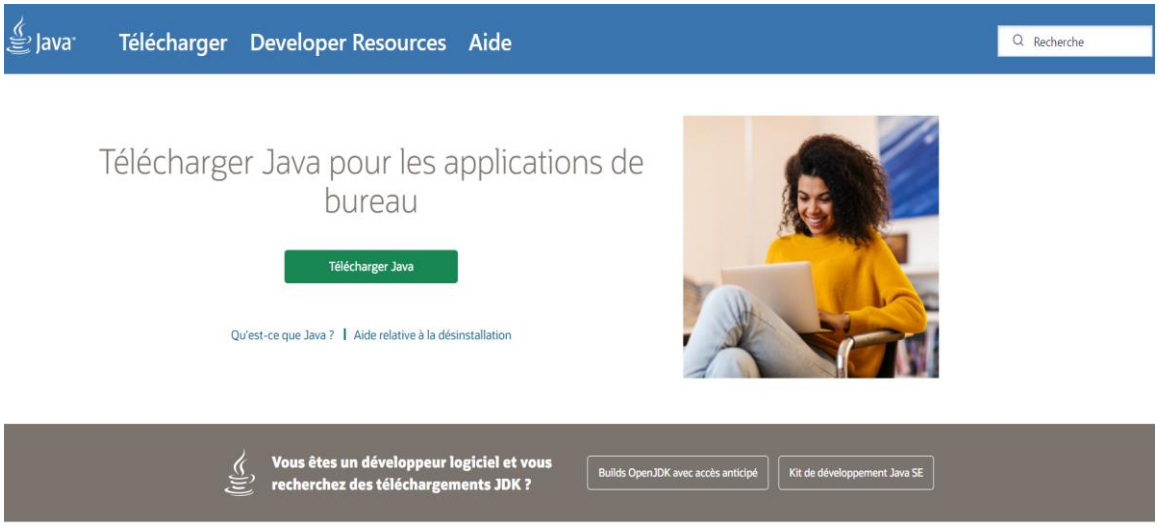


Рисунок 3.3 – Офіційна сторінка завантаження Java

Наступним кроком буде інсталювання середовища розробки Intellij IDEA з порталу JetBrains (рис. 3.4).

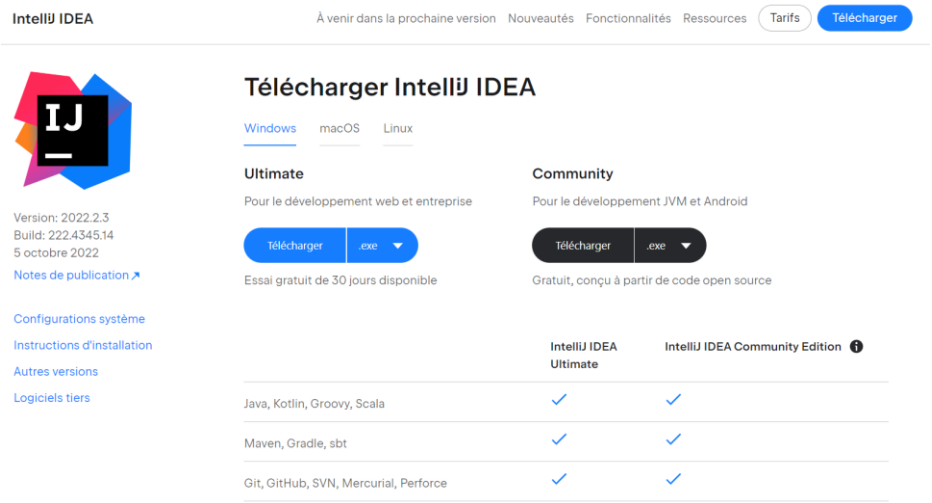


Рисунок 3.4 – Офіційна сторінка завантаження Intellij IDEA

Завантажити та встановити до відповідної дерикторії бібліотеку OpenCV (рис. 3.5).

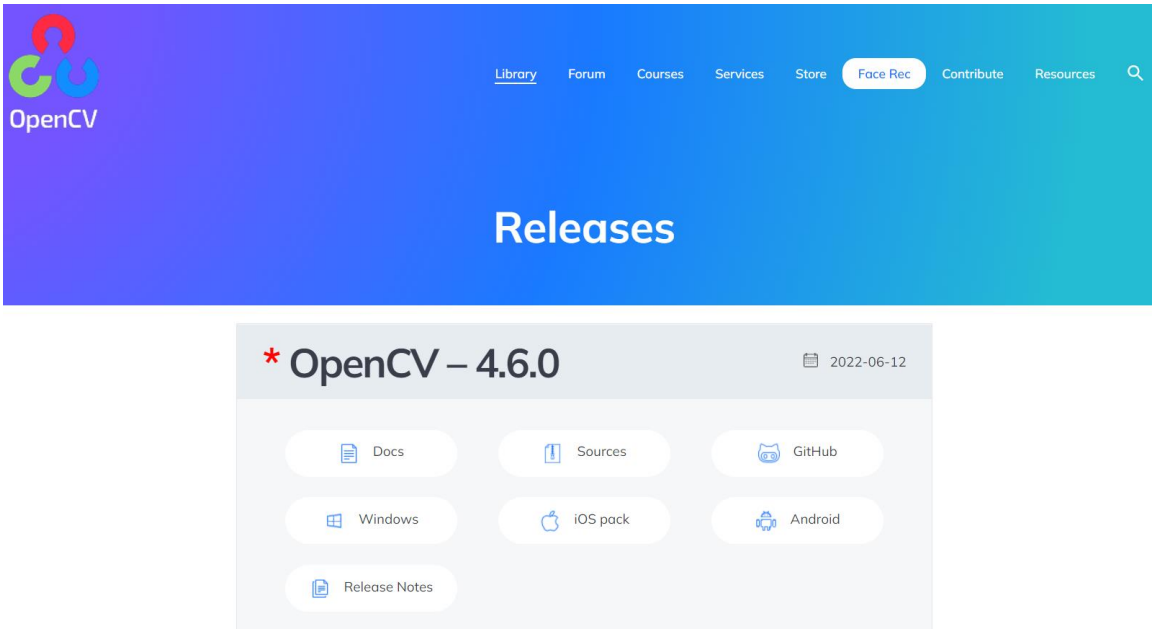


Рисунок 3.5 – Офіційна сторінка завантаження бібліотеки OpenCV

Відкрити файл проекту у середовищі розробки та у розділі налаштувань структури проекту додати необхідні бібліотеки (рис. 3.6, рис. 3.7).

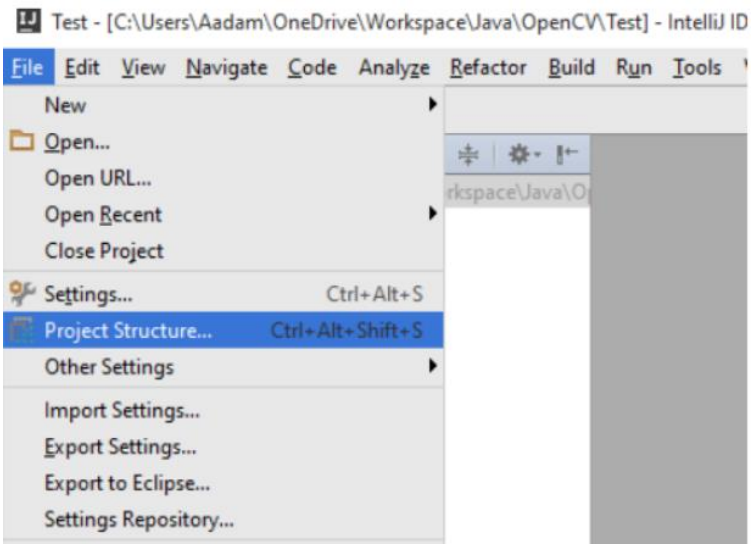


Рисунок 3.6 – Налаштування структури проекту

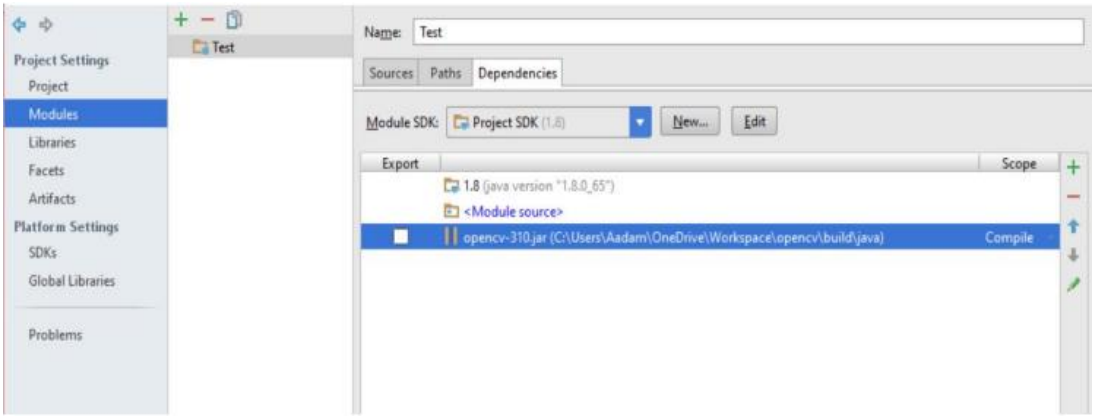
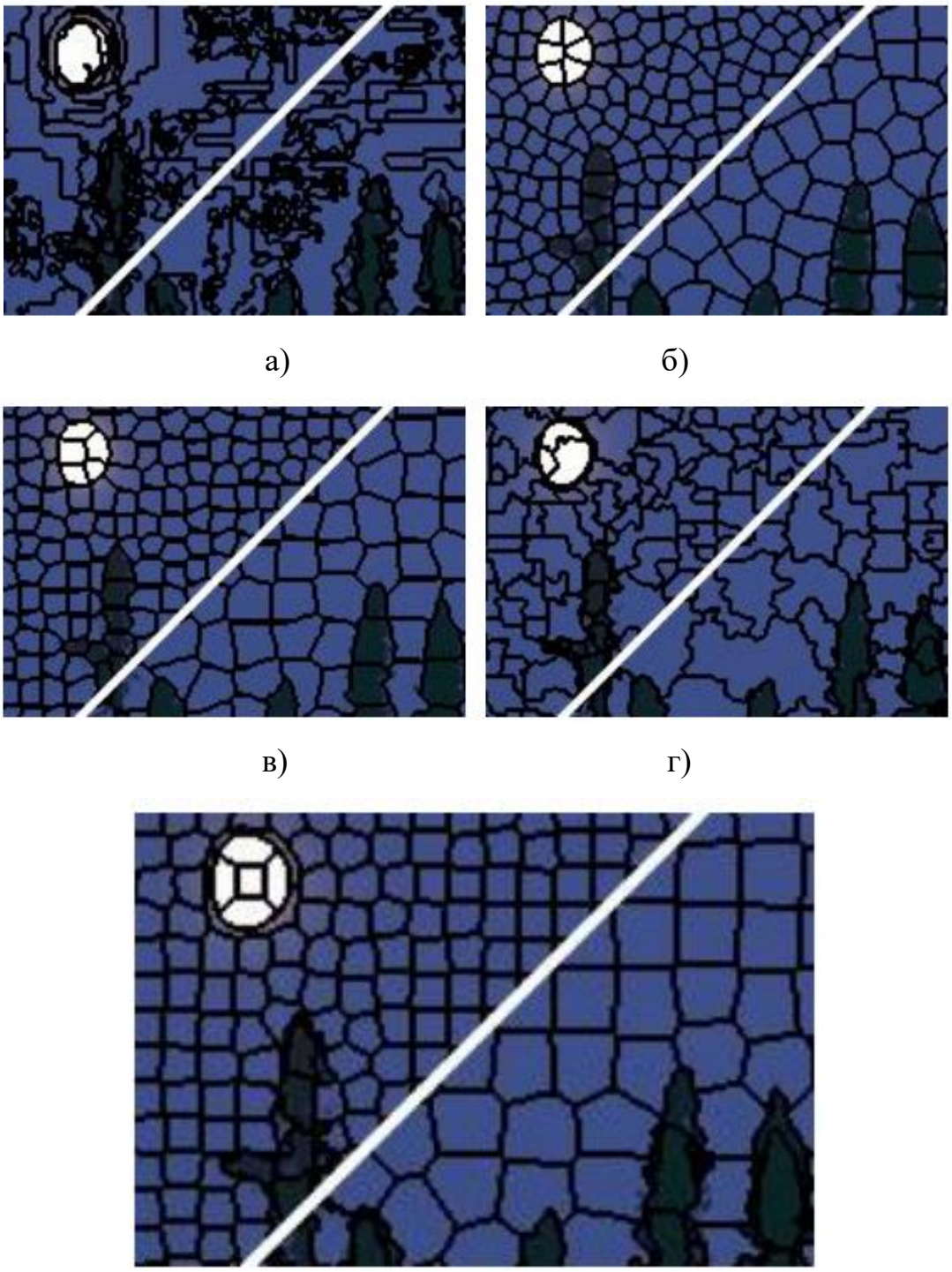


Рисунок 3.7 – Меню додавання бібліотеки OpenCV до структури проекту

У випадку, якщо інструкції з налаштування проекту та середовища були виконані правильно, програма повинна запуститись і коректно вивести результати обробки зображень.

3.5 Тестування розробленої моделі

Порівнюючи підхід SLIC до суперпіксельної сегментації з чотирма найсучаснішими алгоритмами: GS041, NC052, TP093 та QS094, використовуючи їхні вільно доступні вихідні коди. GS04 та NC05 – це алгоритми на основі графів, тоді як TP09 та QS09 – градієнтні методи [33, 34]. NC05 і TP09 призначені для виведення певної кількості суперпікселів, а GS04 і QS09 вимагають модифікації параметрів для отримання такого ж результату. Такий вибір алгоритмів повинен адекватно відображати сучасний рівень розвитку техніки [35, 36]. На рисунку 3.8 зображено візуальний контраст між нашим результатом та різними методами. Щоб надати якісний контраст, ми побудували метрики помилки недосегментації та пригадування границь, використовуючи базову істину сегментації Берклі.



а) б)

в) г)

д)

Рисунок 3.8 – Приклади роботи різних алгоритмів:

а) алгоритм GS; б) алгоритм NC; в) алгоритм TP;

г) алгоритм QS; д) алгоритм SLIC

Помилка недосегментації оцінює неточність, яку допускає алгоритм під час сегментації зображення відносно відомої базової істини. Ця помилка обчислюється в термінах «кровотечі» згенерованих алгоритмом сегментів через базові сегменти [37]. Ця метрика визначає покарання для суперпікселів, які не збігаються з межею сегмента базової істини [37, 38].

Маючи базові сегменти $g_1, g_2, ..., g_M$ та вихідні суперпікселі $s_1, s_2, ..., s_L$, помилка недосегментації для базового сегмента g_i обчислюється наступним чином:

$$U = \frac{1}{N}\left[\sum_{i=1}^{M}\left(\sum_{\left[s_j \mid s_j \cap g_i > B\right]} |S_j|\right) - N\right], \tag{3.1}$$

де $|.|$ – розмір відрізка в пікселях, N – розмір зображення в пікселях, а B – мінімальна кількість пікселів, які потрібно перекрити. Вираз $s_j \cap g_i$ є похибкою перетину або перекриття суперпікселя s_j з по відношенню до сегмента базової істини g_i. В встановлюється на рівні 5% від $|s_j|$, щоб врахувати невеликі помилки в даних сегментації опорної поверхні [39, 40]. Значення U обчислюється для кожного зображення базової істини, а потім усереднюється для отримання графіка на рисунку 3.9.

В якості стандарту міри відкликання границь, що обчислює, яка частка ребер базової істини потрапляє в один піксель принаймні однієї суперпіксельної границі. Граничний відгук кожного з розглянутих методів розглянутих методів від кількості суперпікселів наведено на рисунку 3.10.

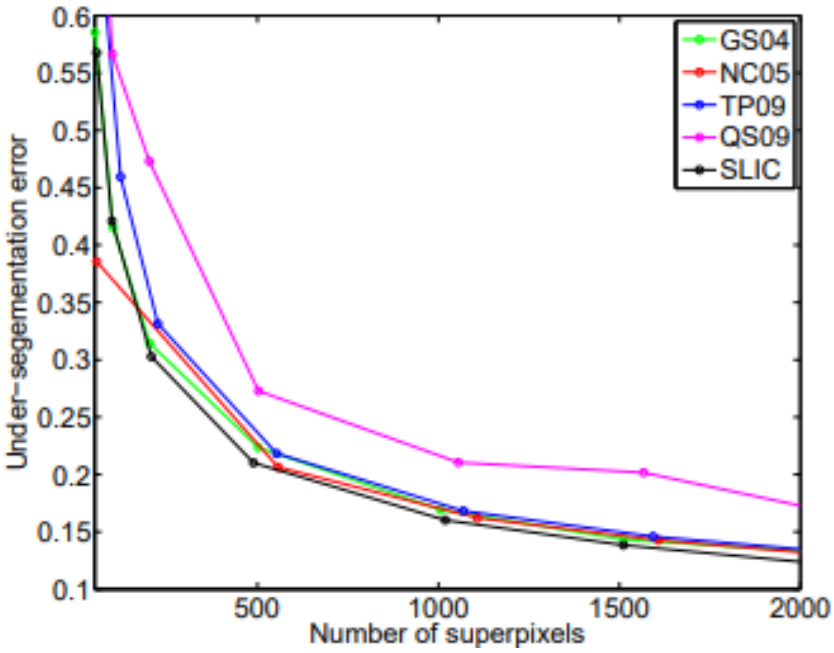


Рисунок 3.9 – Графік залежності похибки недосегментації від кількості суперпікселів

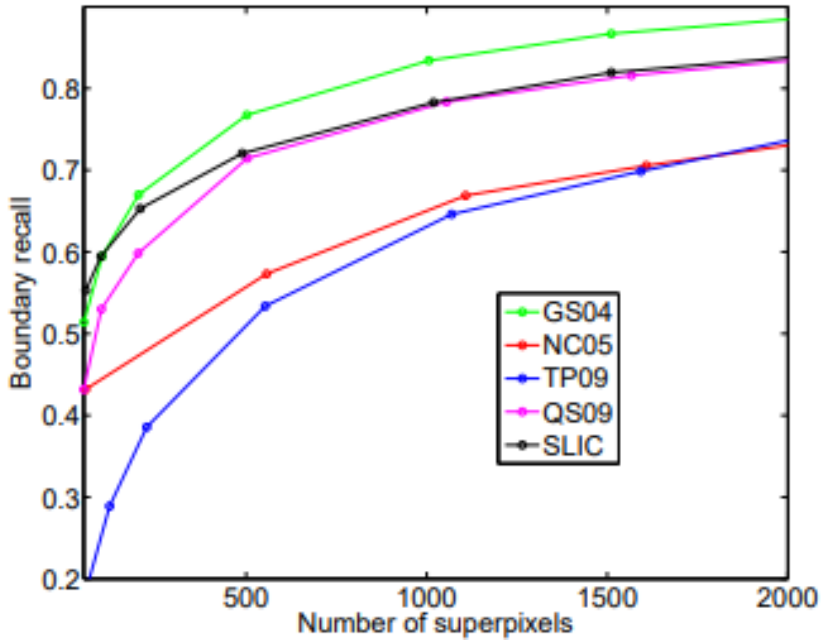


Рисунок 3.10 – Графік відкликання границі від кількості суперпікселів

SLIC більш ніж у 10 разів швидший за TP09 і більш ніж у 500 разів швидший за NC05 для 480 × 320 зображень [41]. Обнадійливо, що він навіть

швидший за GS04 для зображень з більш ніж 500 000 пікселів (рис. 3.11, 3.12). Дана методика завжди має складність *O(N)*, тоді як GS04 має складність *O(NlogN)*.

Це важливо, оскільки навіть найпростіші споживчі цифрові камери створюють зображення з більш ніж 3 мільйонами пікселів [42, 43].

Крім того, GS04 вимагає 5 × *N* пам'яті для запису крайових ваг і порогів, тоді як SLIC вимагає лише 1 × *N* пам'яті (для зберігання відстані між кожним пікселем і його найближчим центром кластера) [44].

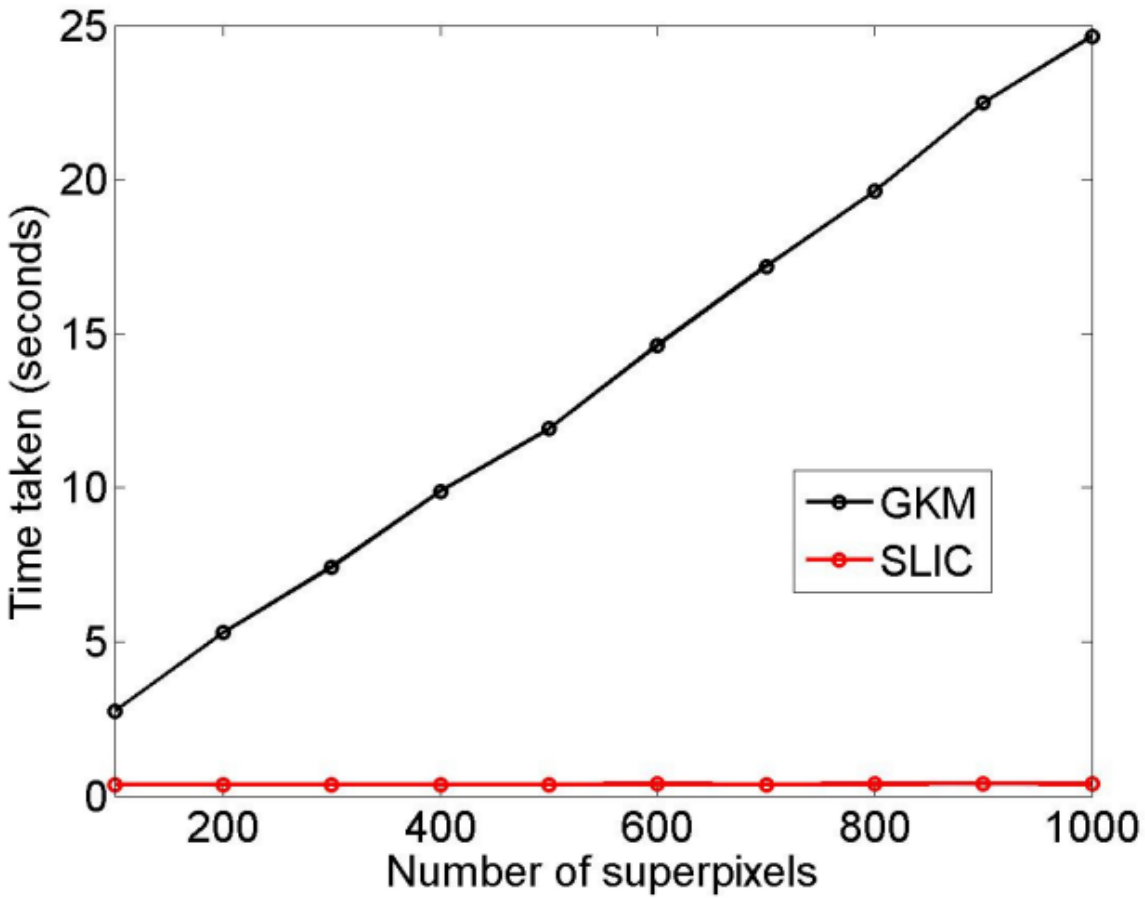


Рисунок 3.11 – Графік залежності часу, витраченого на 10 ітерацій *k*-середніх (GKM) від SLIC алгоритму для різної кількості суперпікселів на вхідних зображеннях розміром 481×321

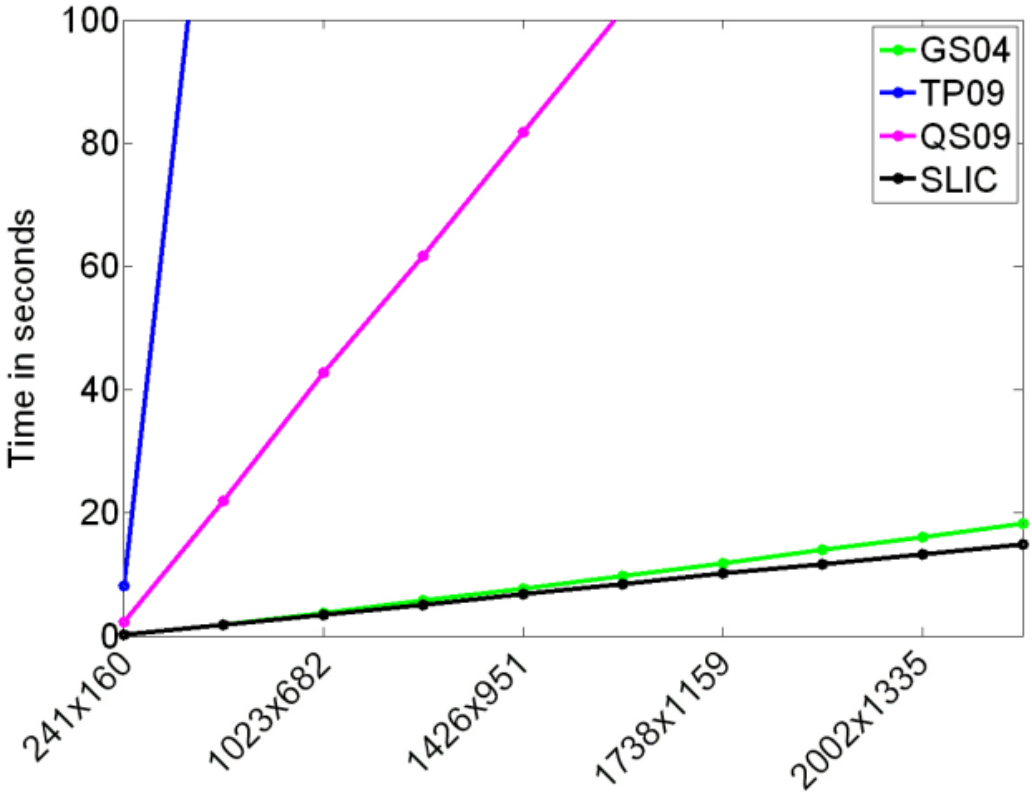


Рисунок 3.12 – Графік залежність сегментації від витраченого часу по відношенню до розміру зображення в пікселях

ВИСНОВКИ

У рамках кваліфікаційної роботи був розроблений та реалізований метод суперпіксельної сегментації зображень.

В даній роботі розглянуто основні методи розробки алгоритмів сегментації та виміряно відносну швидкість і точність декількох архітектур алгоритмів. Використовуючи як статистичні, так і текстурні властивості суперпікселів, найбільш точних результатів, згідно з експериментальними даними, досягає SLIC-метод ресегментації. Випадковий ліс забезпечує найбільш точні результати категоризації серед методів машинного навчання (МН).

Окреслено широкі потреби в основних характеристиках, які швидко виникають під час первинної сегментації. На основі дослідження різноманітних типових задач обробки та ідентифікації зображень представлено систему, що складається з фундаментальних суперпіксельних характеристик. Продемонстровано, як ці фундаментальні характеристики трансформуються в похідні, які безпосередньо використовуються для вирішення певних прикладних ситуацій.

Точність семантичної сегментації становить близько 88,7%. При використанні контекстної інформації (за допомогою CRF-моделі) точність може бути досягнута на рівні 89,8%. Для підвищення швидкості роботи важливо відмовитися від практики розбиття зображення на суперпікселі на користь розбиття на квадратні комірки, використовувати лише статистичні характеристики зображень, а також відмовитися від використання CRF. Крім того, можна зменшити кількість дерев рішень у класифікаторі без суттєвого зниження його точності.

Мету роботи досягнуто.

Результати дослідження апробовано у вигляді тез доповідей під час Міжнародної наукової конференції «MODERN, RELEVANT AND POPULAR RESEARCH OF WORLD SCIENCE» [45].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Rabotiahov, A., Kobylin, O., Dudar, Z., & Lyashenko, V. (2018, February). Bionic image segmentation of cytology samples method. In *2018 14th International Conference on Advanced Trends in Radioelecrtronics, Telecommunications and Computer Engineering (TCSET)* pp. 665–670. IEEE.

2. Работягов, А. В., Ляшенко, В. В., & Кобылин, О. А. (2016). Сегментация сложных изображений цитологических препаратов.

3. Lyashenko, V., Mohammad, A., & Kobylin, O. (2015). Experiments withFusion of Images with Use of Wavelet Transformation in Problems of the Text Information Analysis.

4. Steinley, D. (2006). K-means clustering: a half-century synthesis. *BritishJournal of Mathematical and Statistical Psychology*, *59*(1), 1–34.

5. Huang, Z., & Ng, M. K. (1999). A fuzzy k–modes algorithm for clusteringcategorical data. *IEEE Transactions on Fuzzy Systems*, *7*(4), 446–452.

6. Xu, J., Han, J., Xiong, K., & Nie, F. (2016, July). Robust and Sparse FuzzyK–Means Clustering. In *IJCAI* pp. 2224–2230.

7. Деркач, О. І. (2016). Аналітична обробка текстової інформації за допомогою засобів кластеризації. *Young*, *34*(7).

8. Kobylin, O., Vyskrebentseva, S., & Petrova, R. (2019). Обробка даних, що містять пропуски в задачах кластеризації. *Системи управління, навігації та зв'язку. Збірник наукових праць*, *5*(57).

9. Little, R. J., & Rubin, D. B. (2019). *Statistical analysis with missing data* (Vol. 793). John Wiley & Sons.

10. Jain, A. K., Murty, M. N., & Flynn, P. J. (1999). Data clustering: a review. *ACM computing surveys (CSUR)*, *31*(3), 264–323.

11. Perret, B., Chierchia, G., Cousty, J., Guimarães, S. J. F., Kenmochi, Y., &Najman, L. (2019). Higra: Hierarchical graph analysis. *SoftwareX*, *10*, 100335.

12. Ackermann, M. R. (2009). *Algorithms for the Bregman k–Median problem* (Doctoral dissertation, University of Paderborn).

13. Khachumov, M. V. (2012). Distances, metrics and cluster analysis. *Scientific and Technical Information Processing*, *39*(6), 310–316.

14. Montgomery, D. C., Jennings, C. L., & Kulahci, M. (2015). *Introductionto time series analysis and forecasting*. John Wiley & Sons.

15. Zhang, J., Zhao, Z., Xue, Y., Chen, Z., Ma, X., & Zhou, Q. (2017). Timeseries analysis. *Handbook of Medical Statistics*, *269*.

16. Крашений, І. Е., Попов, А. О., Раміpез, Х., Горріз, Х. М., Крашеный, И. Э., Попов, А. А., ... & Горриз, Х. М. (2016). Використання методів кластеризації в системах нечіткого виводу для діагностики хвороби Альцгеймера на основі ПЕТ–зображень.

17. Штовба, С. Д. (2006). Побудова функцій належності нечітких множин за кластеризацією експериментальних даних. *Інформаційні технології та комп'ютерна інженерія*, (2), 92–95.

18. Bodyanskiy, Y. V., Tyshchenko, O. K., & Mashtalir, S. V. (2019, June). Fuzzy Clustering High–Dimensional Data Using Information Weighting. In *International Conference on Artificial Intelligence and Soft Computing* pp. 385–395. Springer, Cham.

19. Oleg, K., Sergii, M., & Mykhailo, S. (2017, October). Video Clustering via Multidimensional Time–Series Analysis. In *Proceedings of the 9th International Conference on Information Management and Engineering* pp. 60–63. ACM.

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

21. 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.

22. Bodyanskiy, Y., Kobylin, I., Rashkevych, Y., Vynokurova, O., & Peleshko, D. (2018, February). Hybrid fuzzy–clustering algorithm of unevenly and asynchronously spaced time series in computer engineering. In *2018 14th International Conference on Advanced Trends in Radioelecrtronics, Telecommunications and Computer Engineering (TCSET)* pp. 930–935. IEEE.

23. Bodyanskiy, Y., Vynokurova, O., Kobylin, I., & Kobylin, O. (2016). Adaptive fuzzy clustering of short time series with unevenly distributed observations in Data Stream Mining tasks. *Information Technology and Management Science*, *19*(1), 23–28.

24. Женбинг, Х., Бодянский, Е. В., Тыщенко, А. К., & Ткачев, В. Н. (2017). Fuzzy Clustering Data Arrays with Omitted Observations.

25. Kate, R. J. (2016). Using dynamic time warping distances as features for improved time series classification. *Data Mining and Knowledge Discovery*, *30*(2),283–312.

26. Hu, Z., Mashtalir, S. V., Tyshchenko, O. K., & Stolbovyi, M. I. (2018). Clustering matrix sequences based on the iterative dynamic time deformation procedure. *International Journal of Intelligent Systems and Applications*, *10*(7), 66–73.

27. Wang, D., Lu, X., & Rinaldo, A. (2017). DBSCAN: Optimal Rates For Density Based Clustering. *arXiv preprint arXiv:1706.03113*.

28. Lyashenko V., Kobylin O., Selevko O. (2020) Wavelet Analysis and Contrast Modification in the Study of Cell Structures Images. International Journal of Advanced Trends in Computer Science and Engineering. 9(4). – 4701–4706.

29. Bodyanskiy, Y., Shafronenko, A., & Mashtalir, S. (2019, May). Online Robust Fuzzy Clustering of Data with Omissions Using Similarity Measure of Special Type. In *International Scientific Conference "Intellectual Systems of Decision Making and Problem of Computational Intelligence"* pp. 637–646. Springer, Cham.

30. Mashtalir, S. V., Stolbovyi, M. I., & Yakovlev, S. V. (2019). Clustering Video Sequences by the Method of Harmonic k–Means. *Cybernetics and Systems Analysis*, *55*(2), 200–206.

31. Mashtalir, V., Ruban, I., & Levashenko, V. (Eds.). (2019). Advances in Spatio–Temporal Segmentation of Visual Data (Vol. 876). Springer Nature.

32. Kobylin, O., & Lyashenko, V. (2016). Contrast Modification as a Tool toStudy the Structure of Blood Components.

33. Ren, X., & Malik, J. (2003, October). Learning a classification model for segmentation. In *Computer Vision, IEEE International Conference on* (Vol. 2, pp. 10-10). IEEE Computer Society.

34. Miyama, M. (2017, November). Fast stereo matching with super-pixels using one-way check and score filter. In *2017 7th IEEE International Conference on Control System, Computing and Engineering (ICCSCE)* (pp. 278-283). IEEE.

35. Gu, F., Zhang, H., & Wang, C. (2018, July). A classification method for polsar images using SLIC superpixel segmentation and deep convolution neural network. In *IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium* (pp. 6671-6674). IEEE.

36. Kobylin, O. A., Gorokhovatskyi, V. O., Tvoroshenko, I. S., & Peredrii, O. O. (2020). The application of non-parametric statistics methods in image classifiers based on structural description components. Telecommunications and Radio Engineering, 79(10).

37. Kobylin, O., & Lyashenko, V. (2014). Comparison of standard image edge detection techniques and of method based on wavelet transform.

38. Кобилін, О. А., & Творошенко, І. С. (2021). Методи цифрової обробки зображень.

39. Gorokhovatskiy, V. A., Kobylin, O. A., & Kulikov, Y. A. (2015). Application of Granulation of Feature Descriptions in Structural Image Recognition. *Telecommunications and Radio Engineering*, *74*(6).

40. Kuzminska, O., Mazorchuk, M., Morze, N., & Kobylin, O. (2019, June). Digital learning environment of ukrainian universities: The main components to influence the competence of students and teachers. In *International Conference on Information and Communication Technologies in Education, Research, and Industrial Applications* (pp. 210-230). Springer, Cham.

41. Kinoshenko, D., Kobylin, O., Mashtalir, S., & Stolbovyi, M. (2019, March). Metric video retrieval speedup by irrelevant data elimination. In *Eleventh International Conference on Machine Vision (ICMV 2018)* (Vol. 11041, pp. 176-183). SPIE.

42. Бодянский, Е. В., Винокурова, Е. А., Пелешко, Д. Д., Кобылин, И. О., & Кобылин, О. А. (2017). Нечёткая кластеризация временных рядов с неравномерными и асинхронными тактами квантования. *Системи обробки інформації*, (5), 47-54.

43. Lyashenko, V., Matarneh, R., Kobylin, O., & Putyatin, Y. (2016). Contour detection and allocation for cytological images using Wavelet analysis methodology.

44. Lyashenko, V., Kobylin, O., & Ahmad, M. A. (2014). General methodology for implementation of image normalization procedure using its wavelet transform.

45. Усік Д. (2022) Аналіз методу суперпіксельної сегментації зображень, *Abstract of II International Scientific and Practical Conference «Modern, relevant and popular research of world science» (October 04 – 07, 2022). Tokyo, Japan,* pp. 302-304.