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)

ІЄРАРХІЧНА СУПЕРПІКСЕЛЬНА СЕГМЕНТАЦІЯ ЗОБРАЖЕНЬ
(тема)

Виконав:
студент 2 курсу, групи ІНФм-20-1

Горшколєпов А.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)

2021 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)
«____»_____ 2021 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Горшколєпову Антону Вікторовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Ієрархічна суперпіксельная сегментація зображень

затверджена наказом по університету від «_22__»_____10_____2021 року № 1574Ст.

2. Термін подання студентом роботи до екзаменаційної комісії _14___грудня_____ 2021 р.

3. Вихідні дані до роботи Теоретичні відомості про методи обробки зображень; Результати дослідження методу суперпіксельної обробки зображень

4. Перелік питань, що потрібно опрацювати в роботі
1. Огляд основних методів обробки зображень.
2. Аналіз методу сегментації зображень.
3. Дослідження методу суперпіксельної обробки зображень.
4. Реалізація методу суперпіксельної обробки SLIC

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Актуальність проблеми обчислювальної складності під час обробки зображень, постановка задачі реалізації методу суперпіксельної обробки з використанням алгоритму суперпіксельної сегментації SLIC, графічні результати використання методу суперпіксельної обробки 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	Доцент Бєлова Н.В.		

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	22.10.2021	
2	Аналіз завдання, підбір літератури	22.10.21-24.10.21	
3	Аналіз літератури з досліджуваної проблеми	25.10.21-27.10.21	
4	Аналіз технічних засобів	28.10.21-30.10.21	
5	Розробка методу	31.10.21-03.11.21	
6	Програмна реалізація	04.11.21-06.11.21	
7	Оформлення пояснювальної записки	07.11.21-19.11.21	
8	Перевірка на плагіат	14.12.2021	
9	Рецензування	14.12.2021	
10	Підготовка презентації та доповіді	14.12.2021	
11	Занесення роботи в електронний архів	15.12.2021	
12	Попередній захист кваліфікаційної роботи	15.12.2021	

Дата видачі завдання _____ _____ 2021 р.

Студент _____
(підпис)

Керівник роботи _____ _____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 57 с., 34 рис., 45 джерел, 1 таблиця.

ІЄРАРХІЧНА СУПЕРПІКСЕЛЬНА СЕГМЕНТАЦІЯ, АЛГОРИТМ СЕГМЕНТАЦІЇ SLIC, АЛГОРИТМИ ГЕНЕРАЦІЇ СУПЕРПІКСЕЛІВ

Об’єктом дослідження є моделі ієрархічної суперпіксельної сегментації зображень 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.

HIERARCHY SUPERPIXEL SEGMENTATION, SLIC SEGMENTATION ALGORITHM, SUPERPIXEL GENERATION ALGORITHMS

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

The aim of the study 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

ООП – об'єктно-орієнтоване програмування

СС – суперпіксельна сегментація

API – опис способів, якими одна комп’ютерна програма взаємодіє з іншою програмою (Application Programming Interface)

GDI – графічний інтерфейс користувача (Graphics Device Interface)

IDLE – це інтегроване середовище розробки та навчання мовою Python (Integrated Development and Learning Environment)

SLIC – вбудована в BIOS технологія опису ліцензування програмного забезпечення (Software Licensing Description Table)

ВСТУП

Досить часто при аналізі зображень виникає завдання розділення пікселів зображень на групи за деякими ознаками. Такий процес розбиття на групи називається сегментацією. У зв'язку з тим, що в усьому світі набирає обертів застосування інтелектуальних технологій на виробництві або при проведенні досліджень, важливими є методи аналізу зображень. Отже, виникла потреба у впровадженні нових більш складних методів сегментації, що можуть бути застосовані в ряді конкретних завдань.

Суперпіксельні методи широко використовуються на стадії попередньої обробки в якості методів зниження обчислювальної складності за рахунок спрощення зображень при збереженні характеристик зображень у програмах комп’ютерного зору. Зазвичай суперпікселі генеруються на основі значень пікселів, а не з урахуванням характеристик зображення.

Зараз у багатьох сферах, де аналізують відеозображення (геологія, мікробіологія, астрономія тощо), використовують методи сегментації, які підходять саме для цієї окремо взятої сфери діяльності. Хоча ці методи дуже рідко відрізняються чимось суттєвим, оскільки в них подібний принцип роботи. Але є й такі, які відрізняються від основної маси за своїм алгоритмом роботи.

Актуальність дослідження полягає у покращенні роботи алгоритмів ієрархічної сегментації на прикладі алгоритму SLIC.

1 СУЧАСНИЙ СТАН МЕТОДУ СЕГМЕНТАЦІЇ ЗОБРАЖЕНЬ

1.1 Актуальність теми сегментації зображень

Алгоритми оконтурювання меж об'єктів на кольоровому зображенні (алгоритми об'єктної колірної сегментації) є необхідним інструментом для вирішення різних прикладних завдань у галузі обробки кольорових зображень, пов'язаних з їх редагуванням, аналізом, синтезом, відновленням та стисненням. Хоча зараз вже розроблено велика кількість таких алгоритмів, як для автоматичної, так і для контрольованої оператором сегментації, використання більшості з них не забезпечує задовільного результату.

Причина цього в першу чергу черга полягає в тому, що в цих алгоритмах моделлю зображення однорідно забарвленого об'єкта є однорідна за кольором ділянка зображення. Однак, через суттєві неоднорідності потужності та кольоровості освітлення у просторі сцени, а також через складну структури індикатри розсіювання поверхонь об'єктів у сцені, зображення однорідно пофарбованого об'єкта є як правило, суттєво неоднорідним за кольором. Більше того, на зображенні об'єкта з'являються і додаткові контрастні межі (кордони відблисків, тіней та затінків). У результаті, при сегментації зображення класичними методами об’єкт дробиться на дрібніші області, межі яких не відповідають будь-якому стрибка відбивних властивостей поверхні.

Тому, очевидно, що завдання об'єктна сегментації не може бути задовільно вирішена без урахування оптичних явищ, що породжують зображення межі різних типів.

Хоча вже давно зрозуміло, що алгоритми сегментації кольорів можна розвивати, лише орієнтуючись на фізичну модель сцени (з її обмеженнями та наближеннями), число таких алгоритмів, тим більше реалізованих програмно, обчислюється одиницями. У той же час потреба у стійко працюючих алгоритмах колірної сегментації велика. Таким чином, актуальність створення

нових методів та алгоритмів обробки кольорового зображення (що ґрунтуються на фізичному підході) цілком очевидна.

1.2 Визначення сегментації зображень

Сегментація – розбиття зображення на області, що не перетинаються, що покривають усі зображення та однорідні за деякими ознаками: колір, яскравість, текстура тощо. Відомо, що більшість стандартних алгоритмів обробки зображень використовують їх подання у вигляді регулярної піксельної сітки, що обумовлено, більшою ступеня, вихідним способом зберігання зображень у цифровій формі. Однак це не завжди оптимально з багатьох точок зору та, перш за все, для організації наступної обробки.

У цьому плані відносно новим є підхід, заснований на так званії суперпіксельнії сегментації (СС). СС реалізує розбиття зображення на безліч дрібних фрагментів (суперпікселів), що являють собою відносно однорідні групи розташованих поруч пікселів. Кожен суперпіксель потенційно є атомарним регіоном (фрагментом) зображення, тобто всі пікселі, що входять до нього, розглядаються при подальшій обробці як єдине ціле. При цьому суперпікселі не обов'язково повинні мати правильну форму як показано на рисунку 1.1. Природно, завжди є певне кількість помилок, які допускаються при прагненні розбити зображення на однорідні фрагменти.

Рисунок 1.1 – Суперпіксельна сегментації зображення

Як уже зазначалося, суперпіксельна карта зображення, отримана після проведення суперпіксельної сегментації, має низку переваг у порівнянні з звичайною регулярною сіткою пікселів.

По-перше, це обчислювальна ефективність: десятки і сотні тисяч пікселів замінюються лише на десятки або сотні суперпікселів, які виступають в алгоритмах як єдине ціле, що може значно знизити час обробки в алгоритми розпізнавання.

По-друге, при суперпіксельному поданні зображення можна говорити про взаємозв'язки між віддаленими один від одного пікселями, у той час як у піксельному – тільки про зв'язки між поруч пікселів, що лежать.

По-третє, суперпікселі мають більше значення. Кожен суперпіксель є узгодженою одиницею, тому що належать йому пікселі мають схожі кольори, яскравість і інші властивості. Дані властивості суперпікселів зумовлюють їх використання під час вирішення завдань сегментації об'єктів як із відомими, і з невідомими властивостями.

1.3 Приклади застосування сегментації

Деякі практичні застосування сегментації зображень показано на рисунку 1.2:

- медичні зображення;
- виявлення пухлин та інших патологій;
- визначення обсягів тканин;
- хірургія за допомогою комп'ютера;
- діагностика планування лікування;
- вивчення анатомічної структури;
- виділення об'єктів на супутникових знімках;
- розпізнавання осіб;
- розпізнавання відбитків пальців;
- системи керування дорожнім рухом;
- виявлення стоп–сигналів;
- машинний зір;
- розпаралелювання інформаційних потоків при передачі зображень високої роздільної здатності.

Для сегментації зображень було розроблено кілька універсальних алгоритмів та методів. Оскільки загального рішення завдання сегментації зображень немає, часто ці методи доводиться поєднувати зі знаннями з предметної області, щоб ефективно вирішувати це завдання у її предметної області.

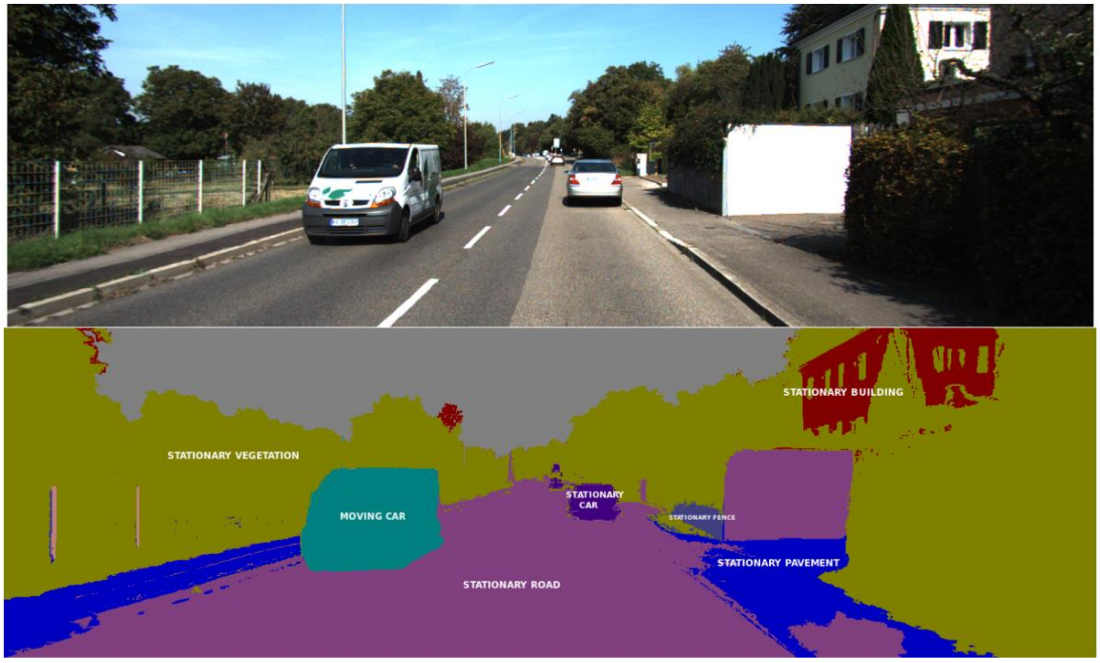


Рисунок 1.2 – Приклад використання сегментації в реальному житті

1.4 Особливості задачі сегментації зображень

Сегментацію зображення можна переформулювати як завдання поділу двовимірного сигналу на зв'язкові області (гладкі в сенсі аналізованих характеристик), що відрізняються від сусідніх областей за своїми яскравими, кольоровими або текстурними характеристиками. Згідно моделі вхідного сигналу слід зазначити такі важливі особливості сегментації зображення:

– зображення є двовимірним сигналом і відрізняються високою просторовою кореляцією характеристик сусідніх елементів;

– області на зображеннях у багатьох випадках мають досить великі розміри, причому протягом областей значення показників можуть помітно змінюватися;

– можлива кількість класів об'єктів на зображеннях, як правило, наперед невідома;

– при сегментації зображення зазвичай не ставиться завдання об'єднання в один клас близьких за характеристиками, але не сусідніх між собою областей.

Через зазначені 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 дуже близькими або навіть перетинатися. До того ж, при поелементному відображенні втрачається інформація про просторове сусідство точок зображення.

Усе це свідчить тому, що сегментація зображення шляхом розв'язання класичної завдання розпізнавання має обмежене застосування. Задовольнити зазначеним особливостям повинен підхід, який використовує критерії близькості як вибраних ознак, так і просторового розташування елементів зображення. Було прийнято ієрархічний підхід, заснований на пірамідальному алгоритмі, що дозволяє порівнювати ознаки просторово сусідніх пар елементів і об'єднувати ті, які в цьому сенсі виявляються близькими.

Згідно з цим алгоритмом зображення подається у вигляді графа, на нижньому рівні якого розташовані елементи зображення, а гілки відповідають окремим об'єктам. Ключовим тут є вибір оцінки близькості сусідніх вершин графа, що в застосування до розглянутої задачі означає вибір простору ознак і способу оцінки близькості точок у цьому просторі. Зіставляти вершини графа, відповідні сусіднім елементам або сегментам зображення, зручно за допомогою відстані просторі ознак, який і буде критерієм при ухваленні рішення про їх злиття або проведення між ними кордону.

1.5 Алгоритми суперпіксельної сегментації зображень

Для сегментації зображень було розроблено декілька універсальних алгоритмів які показано на рисунку 1.3.

Так як спільного рішення для задачі сегментації зображень не існує, часто ці методи доводиться поєднувати зі знаннями з предметної області, для більш ефективного вирішення завдання.

Одна з класифікацій алгоритмів поділяє існуючі методи на наступні підмножини:

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

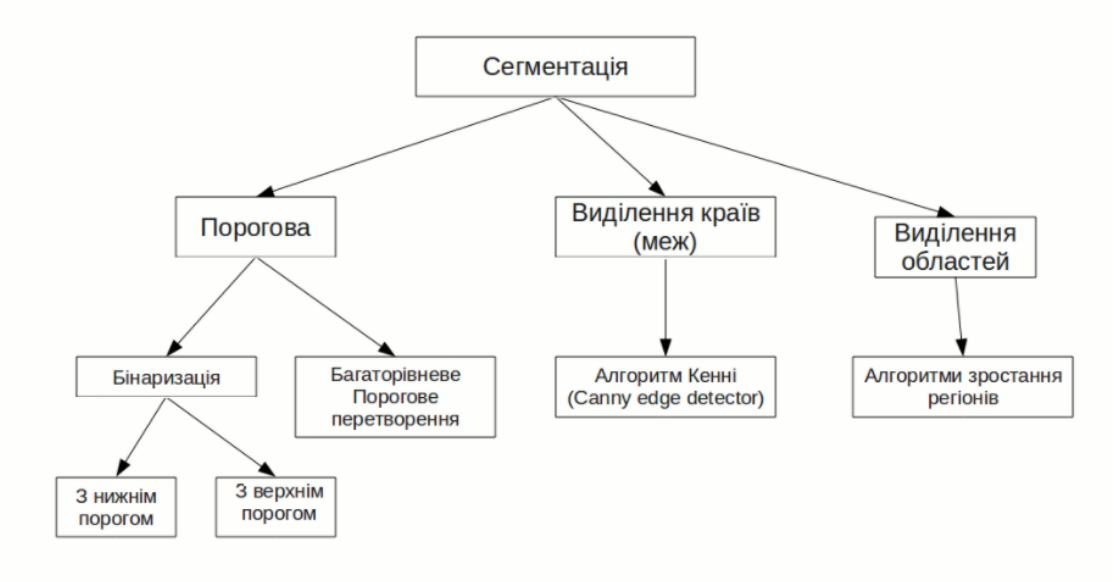


Рисунок 1.3 – Схема класифікації алгоритмів сегментації.

Алгоритми порогової обробки досить прості в реалізації. Поріг – це ознака (властивість), яке допомагає розділити шуканий сигнал на класи. Операція порогового поділу полягає в співставленні функції (наприклад функція може повертати яскравість пікселя) для кожного пікселя зображення з заданим значенням порога.

Алгоритми з даної групи можуть давати неточні результати, особливо при високому рівні шуму на зображенні, що вимагає додаткової перед- або пост-обробки зображення.

Найпростіше перетворення – бінаризація. Вона являє собою операцію порогового поділу, яка в результаті дає бінарне зображення. Метою операції бінаризації є радикальне зменшення кількості інформації, що міститься на зображенні. В процесі бінаризації вихідне напівтонове зображення, що має певну кількість рівнів яскравості, перетворюється в чорно-біле зображення, пікселі якого мають тільки два значення 0 і 1.

Розрізняють декілька видів порогової обробки:

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

Алгоритми виділення країв дозволяють позначити межі елементів на зображенні. Найбільш відомим таким алгоритмом є детектор країв Кенні.

1.6 Оцінка якості алгоритмів сегментації зображень

Якість сегментації зображення може визначатися статистичними, спектральними, характеристиками яскравості зображення. У більшості практичних застосувань якість розглядається як міра близькості двох зображень: реального та ідеального чи перетвореного та вихідного. При такому підході можна оцінювати як суб'єктивну ступінь схожості зображень, так і отримувати об'єктивні оцінки параметрів сигналів зображення:

– моменти першого та другого порядку різницевого сигналу зображень, що порівнюються;

– параметри перетворення як відношення сигнал, коефіцієнти стиснення інформації та інші.

Для проведення порівняльного аналізу методів сегментації використовуються критерії, що ґрунтуються на обчисленні міри відмінності

між результатом сегментації, отриманим за допомогою алгоритму, та сегментом, побудованим експертом на основі візуального аналізу зображення.

Нижче наведено критерії оцінки якості сегментації:

– показник надмірної сегментації;
– показник недостатньої сегментації;
– загальна помилка сегментації.

У результаті сегментації 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 при оцінці якості сегментації зображень вибрано два критерії: критерій, що показує ступінь подібності сегментованого та ідеального контурного зображень (FOM).

Суб'єктивні критерії – це критерії візуального сприйняття, оцінювані у процесі експертизи певною групою спостерігачів (експертів). Найбільшого поширення набув метод оцінок, у якому спостерігач оцінює якість зображення у балах за певною шкалою, вважаючи, що ідеальне зображення має максимальний бал як показано у таблиці 1.

Таблиця 1 Шкала суб’єктивних оцінок якості зображення

Нормалізована	П'ятибальна		Семибальна шкала погіршення
	Шкала якості	Шкала погіршення	
1	5(відмінно)	5 (непомітно)	1 (помітно)
0,875	-	-	2(ледве помітно)
0,75	4(добре)	4(помітно, але не заважає)	3(погіршує але не заважає)
0,625	-	-	4(погіршує але не заважає)

Продовження таблиці 1

0,5	3(задовільно)	3(помітно, трохи заважає)	5(трохи заважає)
0,25	2(погано)	2(заважає)	6(заважає)
0	1(дуже погано)	1(сильно заважає)	7(дуже заважає)

1.7 Постановка задачі дослідження

За умовами поставленого завдання потрібно дослідити методи, який буде розпізнавати та сегментувати зображення. Проєктування та розробку потрібно зробити відповідно до поставлених вимог та поставлених строків.

Таким чином, можна зробити висновок, що сегментація є актуальним завданням для обробки і розпізнавання зображень. Тому ставиться завдання розробки алгоритму суперпіксельної обробки, а саме поділу зображення на групи пов’язаних областей, які є однорідними і не перекриваються (група пікселів, які також називається суперпікселями).

Метою дослідження є розробка методів, що базуються на сегментуванні суперпікселів.

Для досягнення мети необхідно вирішити такі завдання:

– провести аналіз існуючих методів сегментування зображень;

– розробити алгоритм сегментації суперпікселей;

– реалізувати комп’ютерну модель для ієрархічного суперпіксельного сегментування.

2 МАТЕМАТИЧНІ МОДЕЛІ АЛГОРИТМІВ СЕГМЕНТАЦІЇ ЗОБРАЖЕНЬ

2.1 Алгоритми суперпікселей

2.1.1 Графічні алгоритми

Графічний алгоритм Метод генерації суперпікселів на основі графа розглядає кожен піксель як вузол на графіці. Вага краю між двома вузлами пропорційна схожості між сусідніми пікселями. Суперпікселі створюються шляхом мінімізації функції вартості, визначеної на графіку.

NC05 – нормалізований алгоритм. Рекурсивно використовує мітки контуру та текстури для сегментації графіки всіх пікселів у зображенні, тим самим глобально мінімізуючи функцію вартості, що визначається краєм на межі сегментації як показано на рисунку 2.1. Він виготовляє дуже рівні, візуально приємні суперпікселі. Однак NC05 має відносно погане зчеплення з кордонами, і це найповільніший метод (особливо для великих зображень), незважаючи на існування алгоритмів, що намагаються прискорити.

Рисунок 2.1 – Робота алгоритму NC05

GS04-Felzenszwalb та Huttenlocher запропонували альтернативний метод, заснований на графіку, який застосовувався для генерації суперпікселів. Він приймає пікселі як вузли графа, так що кожен суперпіксель є найменшим кістяком пікселів, що й демонструється на рисунку 2.2.

На практиці GS04 добре дотримується межі зображення, але виробляє суперпікселі дуже нестандартних розмірів та форм. Однак він не забезпечує чіткого контролю над кількістю суперпікселів або їх компактністю.

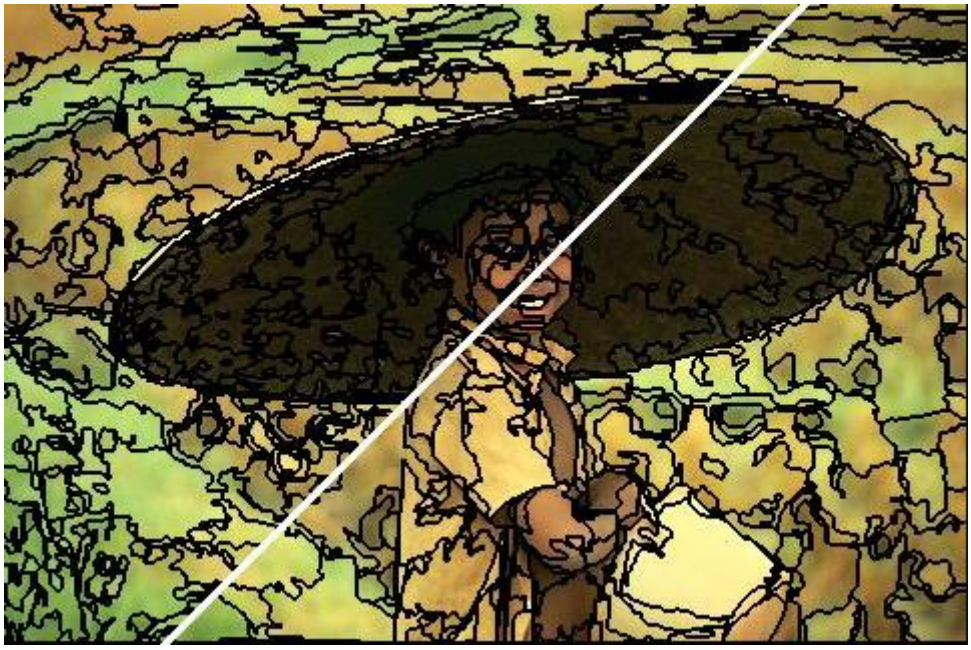


Рисунок 2.2 – Робота алгоритму GS04

SL08-Муритал запропонував метод генерації суперпікселів, відповідних сітці, шляхом визначення найкращого шляху або стику для поділу зображення на дрібніші вертикальні або горизонтальні області. Використаємо метод вирізання графіка, аналогічний SeamCarving, щоб знайти найкращий шлях.

GCa10 і GCb10 – використовує метод глобальної оптимізації, аналогічний роботі із синтезу текстур. Супер пікселі виходять шляхом зшивання блоків зображення, що перекриваються, так що кожен піксель належить тільки одній з областей, що перекриваються. Існує два варіанти цього методу: один для створення компактних суперпікселів (GCa10) та один для суперпікселів постійної інтенсивності (GCb10).

2.1.2 Методи на основі градієнтного підйому

Починаючи з грубої початкової кластеризації пікселів, метод градієнтного підйому ітеративно змінює кластери доти, доки не будуть виконані деякі критерії збіжності для формування суперпікселів.

MS02-InB середній зсув, ітеративний процес пошуку шаблону, використовуваний визначення локального максимуму функції щільності, застосовується до першого шаблону у просторі ознак кольору чи інтенсивності зображення. Пікселі, що сходяться до одного й того самого шаблону, визначають суперпікселі. MS02 - це старіший метод, що дозволяє отримувати суперпікселі неправильної форми та нерівномірного розміру. Це складність O(N^2) O(N2) робить його відносно повільним і не дає прямого контролю над кількістю, розміром або компактністю суперпікселів.

QS08-Fast shift також використовує пошук шаблону для схем сегментації. Він використовує процес зсуву medoid для ініціалізації сегментації. Потім перемістіть точку пошуку у просторі ознак до найближчого сусіда, збільшивши оцінку щільності Парзена. Незважаючи на те, що QS08 має відносно гарне зчеплення з межами, швидкість роботи досить низька та складна. А QS08 не дозволяє явно контролювати розмір чи кількість супер пікселів. У попередніх роботах використовувалося позиціонування об'єкта QS08 та сегментація руху.

WS91-Метод вододілу починається з локального мінімуму і виконує градієнтний підйом для створення вододілів та окремих ліній водозбору. Суперпікселі, що отримуються в результаті, зазвичай мають дуже неправильний розмір і форму і не володіють хорошою адгезією на кордонах. Метод відносно швидкий, але не дозволяє контролювати кількість суперпікселів або їх компактність.

TP09-Турбопіксельний метод використовує геометричний потік, заснований на заданому рівні, для поступового розширення набору вихідних позицій. Геометричний потік залежить від локального градієнта зображення

для регулярного розподілу суперпікселів на площині зображення. На відміну від WS91, суперпікселі TP09 повинні мати однаковий розмір, компактність та прилягання до країв. TP09 спирається на алгоритми різної складності, але на практиці, приблизно O(N) O(N). Складність – один із найповільніших із розглянутих алгоритмів, і він демонструє відносно погану граничну адгезію.

2.1.3 Алгоритми генерації суперпікселів

Основні алгоритми генерації суперпікселів. Після опису кожного з них запропоновані приклади трьох варіантів суперпіксельної сегментації та приклад схеми обробки пікселів при сегментації як показано на рисунку 2.3, недосегментації, при якій кількість результуючих суперпікселів незначна та їх границі зазвичай значно відрізняються від границь об’єктів на зображенні, оптимальної сегментації(яку дозволяє провести алгоритм), коли досягається компроміс між кількістю, розміром суперпікселів та відтворенням границь об’єктів, пересегментації, при якій кількість суперпікселів надзвичайно велика, а їх розмір наближується до розміру окремих пікселів.

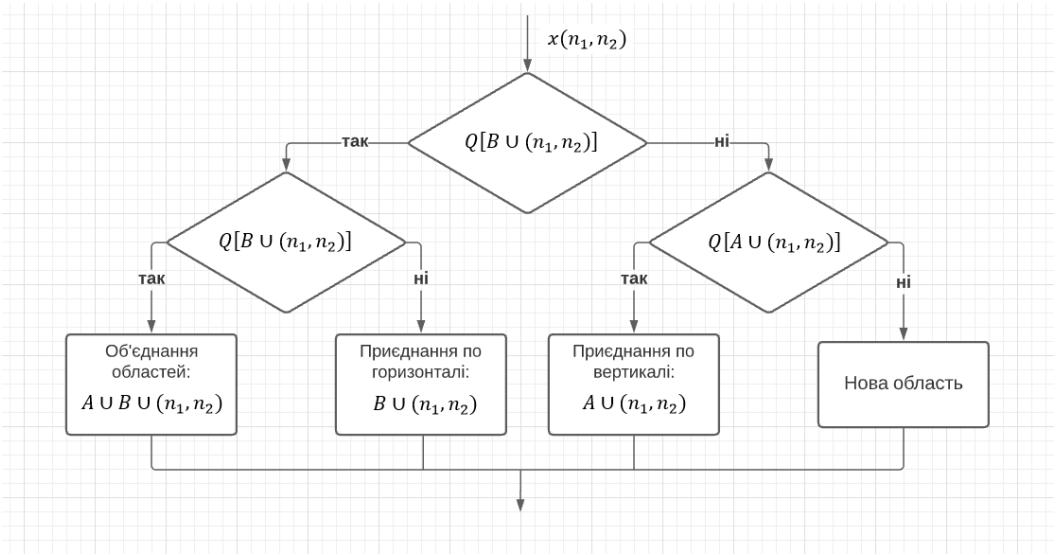


Рисунок 2.3 – Схема обробки кожного пікселя при сегментації зображення

SLIC – адаптація методу кластеризації k-середніх для генерації суперпікселів. Збільшення швидкодії порівняно із класичним методом *k*-середніх досягається за рахунок зменшення регіону пошуку для кожного центра кластера, що призводить до зменшення кількості розрахунків дистанції. Таким чином, пошук схожих пікселів проводиться в регіоні 2S × 2S навколо центра кластера, де $S = \sqrt{N/k}$, N – загальна кількість пікселів на зображенні, k – кількість кластерів. Розрахунок дистанції D між окремим пікселем та центром кластеру виконується за формулою:

$$D = \sqrt{d_c^2 + \left(\frac{d_s}{S}\right)^2 m^2}, \tag{2.1}$$

де d_c – колірна дистанція:

$$d_c = \sqrt{(l_j - l_i)^2 + (a_j - a_i)^2 + (b_j - b_i)^2}, \tag{2.2}$$

d_s – просторова дистанція:

$$d_s = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}, \tag{2.3}$$

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

Коли значення m велике, просторова близькість більш важлива та результуючі сеперпікселі більш компактні. І навпаки, коли значення m мале, важливіша колірна схожість та результуючі суперпікселі краще притримуються меж об’єктів, однак мають менш регулярну форму та розмір, що демонструється на рисунку 2.4.

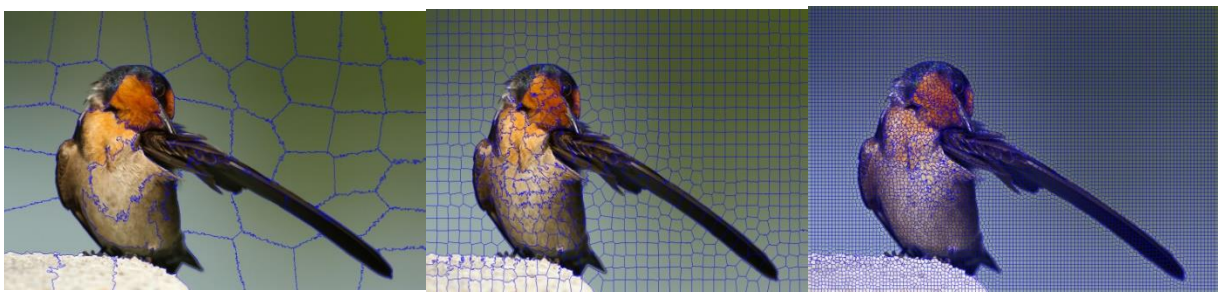


Рисунок 2.4 – Приклад роботи алгоритму SLIC

SLICO – версія класичного SLIC без додаткових параметрів, що регулюють компактність продемонстровано на рисунку 2.5. В цьому алгоритмі фактор компактності розраховується для кожного кластера окремо. А саме, параметр m обирається як максимальна колірна відстань в межах кластеру, що отримана на попередній ітерації.

Рисунок 2.5 – Приклад роботи алгоритму SLICO

Watershed – алгоритм, що має в своїй основі ідею започатковану з географії. Він базується на представленні зображення у вигляді топографічного рельєфу, де значення кожного елементу зображення характеризує його висоту в заданій точці(на зображені значення «висоти» зазвичай відповідає значенню градієнта).

По аналогії з процесами у природі, вода заповнює спочатку найнижчі зони, поступово підіймаючись в гору та утворюючи водозбірні басейни. В місцях, де вода з різних басейнів об’єднується, формується границя водорозділу. На зображенні ця границя буде представляти границю отриманого кластеру. Існують різноманітні реалізації алгоритму. Варіант запропонований Ф. Мейером для симуляції описаного вище процесу

застосовує впорядковану множина черг з пріоритетом. Робота алгоритму розпочинається з виявлення точок локального мінімуму, присвоєнні їм унікальних відміток, та внесення до черги зображено на рисунку 2.6.

Надалі з черги видаляється найпріорітетніший елемент, його сусідам, що не мають позначок, назначається мітка видаленого елемента та виконується їх занесення до черги в залежності від пріоритету. Попередній шаг виконується доки з черги не будуть видалені всі елементи.

Рисунок 2.6 – Робота алгоритму Watershed

Compact Watershed – розширення алгоритму, запропонованого Ф. Мейером, для генерації суперпікселів, що мають більш однорідну форму та розмір зображені на рисунку 2.7. Даний алгоритм відрізняється від свого пращура розподіленням початкових елементів з позначками на регулярній сітці, а не в зонах локального мінімуму, що призводить до більш однорідного розподілення результуючих сегментів. Крім того, тут вноситься обмеження компактності, завдяки зміні критерію пріоритетності в черзі, що дозволяє контролювати розмір та форму результуючих суперпікселів.

Рисунок 2.7 – Робота алгоритму Compact Watershed

FH(Felzenszwalb and Huttenlocher) – алгоритм, що базується на представленні зображення у вигляді графа. Вага ребер, що поєднують сусідні пікселі(вершини), обчислюється як різниця інтенсивності:

$$w((v_i, v_j)) = |I(p_i) - I(p_j).\quad (2.4)$$

Основна ідея базується на побудові таких сукупностей зв'язних вершин (кластерів), щоб вага ребер в середині кластеру була меншою порівняно з вагою ребер, що поєднують вершини із різних кластерів продемонстровано на рисунку 2.8.

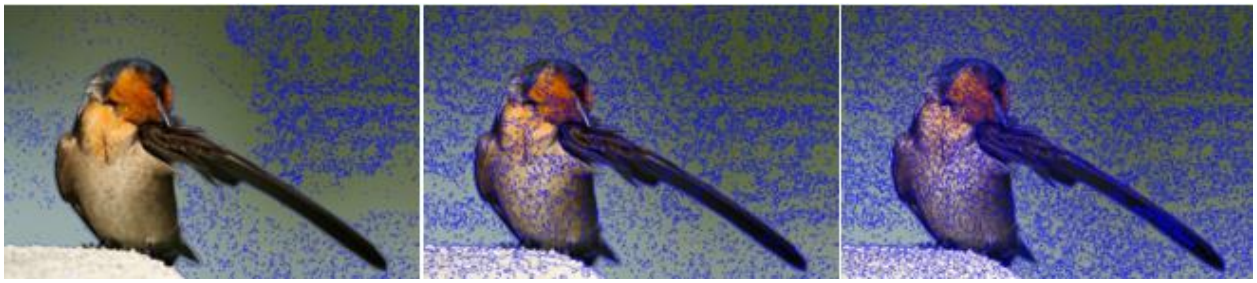


Рисунок 2.8 – Робота алгоритму FH

Таким чином вводяться поняття внутрішньої різності $\mathrm{Int(C)}$ та різності між двома компонентами $\mathrm{Dif(C1, C2)}$:

$$Int(C) = \max_{e \in MST(C,E)} w(e),\quad (2.5)$$

де $w(e)$ – вага ребра e,

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

$$Dif(C1, C2) = \min_{v_i \in C_1, v_j \in C_2, (v_i, v_j) \in E} w((v_i, v_j)),\quad (2.6)$$

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

Надалі вводиться предикат наявності границі між компонентами(кластерами) $D(C1, C2)$:

$$D(C1, C2) = \begin{cases} true\ if\ Dif(C1, C2) > MInt(C1, C2) \\ false\ otherwise \end{cases}, \quad (2.7)$$

де мінімальна внутрішня різність MInt визначається за формулою:

$$MInt(C1, C2) = min(Int(C1) + \tau(C1), Int(C2) + \tau(C2)), \quad (2.8)$$

де порогова функція τ контролює рівень на який різниця між двома компонентами має бути більшою порівняно з їхніми внутрішніми різностями, щоб можна було провести границю між ними. Для зменшення помилок у малих за розміром компонентах, використовується наступна формула:

$$\tau(C) = k/|C|, \quad (2.9)$$

де $|C|$ визначає розмір компонента *C,*

k – деяка константа, яку встановлюватиме користувач.

За допомогою k можна опосередковано впливати на розмір результуючих суперпікселів.

Quick Shift – алгоритм, що відноситься до сімейства алгоритмів пошуку моди. Алгоритм продемонстровано на рисунку 2.9.Він базується на апроксимації ядерного зсуву середнього (mean-shift). Алгоритм зсуває кожну точку x_i до найближчого сусіда, у якого щільність P більша, утворюючи тим самим траєкторію y:

$$y_i(1) = \underset{j:P_j>P_i}{\operatorname{argmin}} D_{ij}, \quad P_i = \frac{1}{N}\sum_{j=1}^{N} \phi(D_{ij}). \quad (2.10)$$

Алгоритм об’єднує всі точки в одне дерево, в якому ребра, що перевищують поріг τ розбиваються, утворюючи кластери. Щільність може розраховуватися по-різному, одним із варіантів є використання ізотропного ядра Гаусса:

$$P(x) = \frac{1}{2\pi\sigma^2 N}\sum_{i=1}^{N} e^{\frac{-||x-x_i||^2}{2\sigma^2}}. \quad (2.11)$$

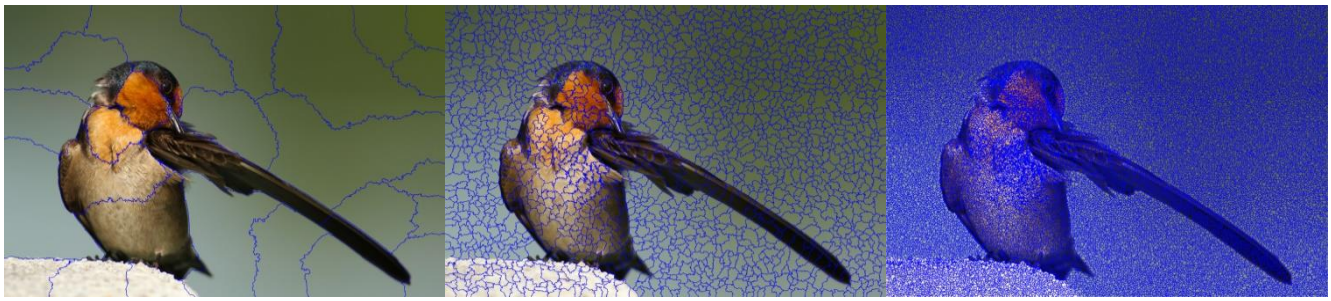


Рисунок 2.9 – Робота алгоритму Quick Shift

SEEDS – алгоритм суперпіксельної сегментації, що базується на ідеї максимізації енергії. Цільова енергетична функція E розбиття на суперпікселі s розраховується як сума двох величин. Перша величина H(s) характеризує однорідність кольору в рамках суперпікселів, друга величина G(s) є опціональною та характеризує однорідність границь суперпікселів:

$$E(s) = H(s) + \gamma G(s), \quad (2.12)$$

$$H(s) = \sum_k \Psi(c_{A_k}), \quad (2.13)$$

$$c_{A_k}(j) = \frac{1}{Z}\sum_{i\in A_k} \delta(I(i) \in Hj), \quad (2.14)$$

$$\Psi\left(c_{A_k}\right) = \sum_{\{H_j\}}(c_{A_k}(j))^2, \quad (2.15)$$

де $\Psi\left(\mathrm{c}_{\mathrm{A_k}}\right)$ – функція, що є мірою якості розподілення кольору в межах кластеру k,

$\mathrm{c}_{\mathrm{A_k}}$ визначає гістограму кольору множини пікселів $\mathrm{A_k}$, що утворюють кластер k,

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

Для розрахунку величини G(s) навколо кожного пікселя i будується патч $\mathrm{N_i}$. По аналогії з величиною розподілення кольору, друга величина розраховується на основі гістограми міток суперпікселів, що присутні в межах патчу:

$$b_{N_i}(k) = \frac{1}{Z}\sum_{j \in N_i} \delta(j \in A_k), \quad (2.16)$$

$$G(s) = \sum_i \sum_k (b_{N_i}(k))^2. \quad (2.17)$$

Таким чином, енергетична функція приймає більші значення при більш однорідних сегментах у кольоровому просторі, що мають гладкі границі. Оптимізація цієї функції відбувається за рахунок перенесення граничних блоків пікселів або окремих пікселів до сусідніх сегментів, у випадку, якщо таке перенесення призводить до збільшення значення E(s). Відповідне уточнення границь починається з початкової сегментації на однорідній сітці та продовжується з поступовим зменшенням блоків пікселів, що переходять з одного кластеру в інший, що показано на рисунку 2.10.

Рисунок 2.10 – Приклад роботи алгоритму SEEDS

LSC (Linear Spectral Clustering Superpixel) – алгоритм, що базується на зв'язку між цільовими функціями нормалізованих зрізів F_{Ncuts} та зваженого методу k-середніх F_{k-m}:

$$F_{k-m} = \sum_{k=1}^{K} \sum_{p \in \pi_k} w(p) \parallel \phi(p) - mk \parallel^2, \tag{2.18}$$

$$m_k = \frac{\sum_{q \in \pi_k} w(q)\phi(q)}{\sum_{q \in \pi_k} w(q)}, \tag{2.19}$$

$$F_{\text{Ncuts}} = \frac{1}{K} \sum_{k=1}^{K} \frac{\sum_{p \in \pi_k} \sum_{q \in \pi_k} W(p,q)}{\sum_{p \in \pi_k} \sum_{q \in V} W(p,q)}, \tag{2.20}$$

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 простором ознак,

W(p, q) – близькість між двома точками p та q,

V – множина всіх вершин графу(пікселів).

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:

$$\phi(p) = \frac{1}{w(p)}\left(C_c \cos\frac{\pi}{2} l_p, \sin\frac{\pi}{2} l_p, 2.55C_c \cos\frac{\pi}{2}\alpha_p, 2.55C_c \sin\frac{\pi}{2}\alpha_p 2.55C_{c\cos\frac{\pi}{2}\beta_p}\right.$$

$$\left. 2.55C_c \sin\frac{\pi}{2}\beta_p, C_s \cos\frac{\pi}{2} x_p, C_s \sin\frac{\pi}{2} x_p, C_s \cos\frac{\pi}{2} y_p, C_s \sin\frac{\pi}{2} y_p\right), \quad (2.21)$$

$$w(p) = \sum_{q\in V} W(p,q) = w(p)\phi(p) \cdot \sum_{q\in V} w(q)\,\phi(q), \quad (2.22)$$

де C_c та C_s використовуються для контролю відносної важливості колірної та просторової інформації відповідно. Алгоритм продемонстровано на рисунку 2.11.

Рисунок 2.11 – Робота алгоритму LSC

2.2 Порівняння відомих алгоритмів

У ході виконаного аналізу для порівняння різних підходів до суперпіксельної сегментації було обрано три сучасні та ефективних алгоритму з кожної групи (розділяючі, що об'єднують, дискримінаційні):

– Алгоритм Калініна;

– Алгоритм ERS;

– Алгоритм SLIC.

Ілюстративні приклади роботи даних алгоритмів представлені на рисунку 2.12

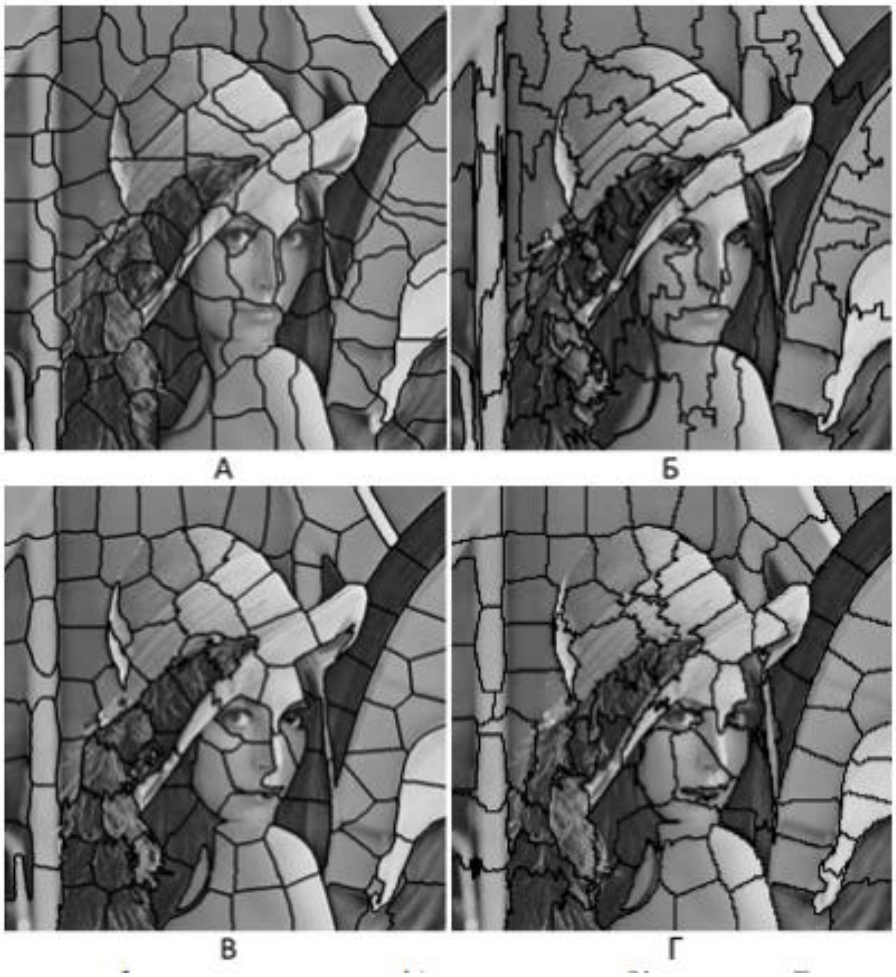


Рисунок 2.12 – Приклади роботи алгоритмів:
А – алгоритм Калініна, Б – алгоритм ERS, В – алгоритм SLIC, Г – модифікований алгоритм SLIC

Інструментарій для дослідження та порівняння алгоритмів реалізовано мовою Matlab. Аналіз проводився на основі зображень із тестового набору Каліфорнійського університету в Берклі. Всі алгоритми реалізовані мовою програмування C++ із відповідним підключенням до середовища Matlab. На рисунку 2.13 наведено порівняння точності алгоритмів. Найбільший відсоток виділених меж має алгоритм ERS, на той час як алгоритм Калініна має найменшу надмірність сегментації.

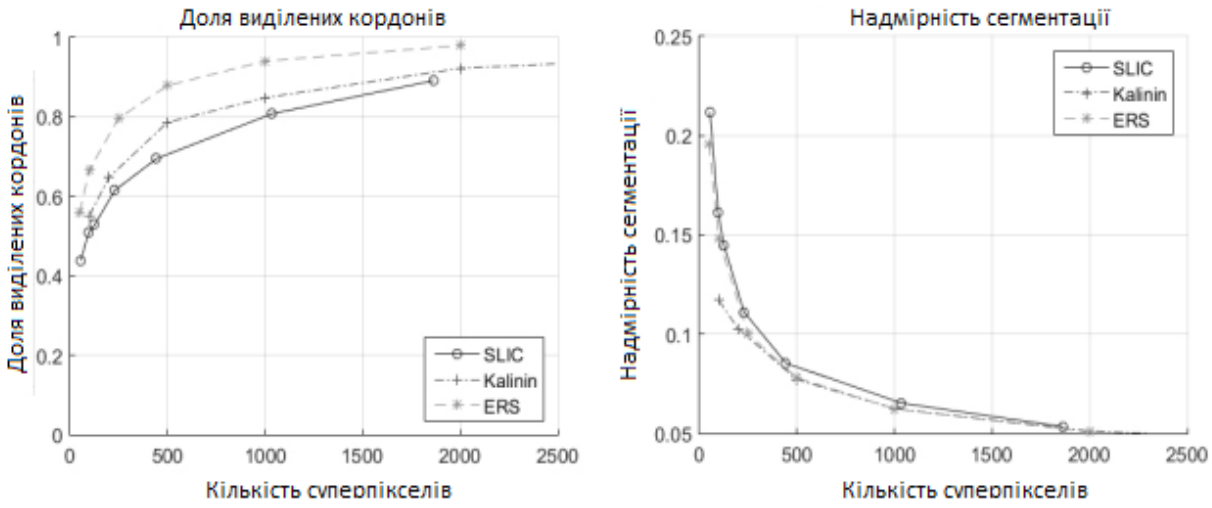


Рисунок 2.13 – Порівняння точності алгоритмів (ліворуч – відсоток виявлених меж, праворуч – надмірність сегментації)

З рисунку 2.14 видно, що SLIC перевершує інші алгоритми за часом роботи та компактності одержуваних суперпікселів.

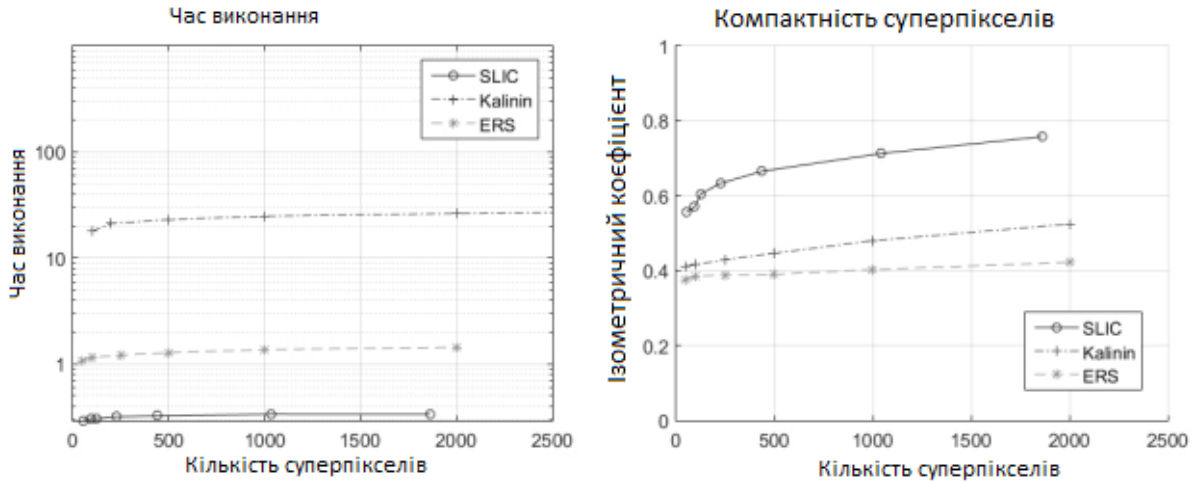


Рисунок 2.14 – Порівняння часу роботи та компактності алгоритмів (ліворуч – час роботи, праворуч – ізодіаметричний коефіцієнт)

Збільшення числа змінених пікселів при шумах типу «сіль та перець» мало позначається на результатах алгоритмів Калініна та ERS, але дуже погіршує результати алгоритму SLIC, що показано на рисунку 2.15.

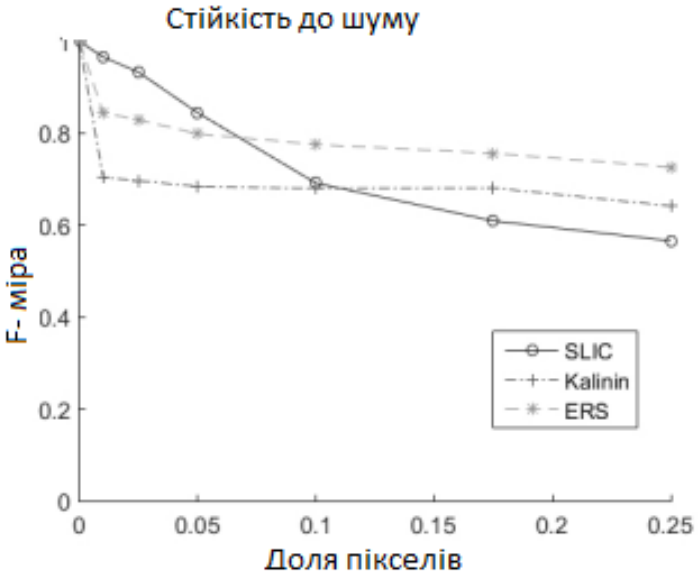


Рисунок 2.15 – Порівняння стійкості алгоритмів до шуму типу «сіль та перець»

Таким чином, жоден з трьох розглянутих алгоритмів не перевершує решту за всіма критеріями, кожен з них має свої переваги та недоліки. Тому далі для дослідження можливостей розвитку методів суперпіксельної сегментації, спрямованого забезпечення сегментації зображень, що містять фрагменти з однаковою яскравістю та неоднорідною текстурою було обрано алгоритм SLIC. Це пов'язано з тим, що алгоритм SLIC досить простий у реалізації і водночас ефективний, а його висока швидкість роботи спрощує проведення експериментів.

2.3 Модифікація алгоритму SLIC для аналізу зображень з різною текстурою

Алгоритми суперпіксельної сегментації зображень мають великі можливості для розвитку та модифікації. Одним із можливих підходів до їх удосконалення є залучення додаткових ознак до сегментації з метою покращення загальної якості СС. Розглянемо Класичний алгоритм SLIC. Як уже зазначалося, алгоритм є просторово орієнтованою реалізацією алгоритму кластеризації k-середніх. Однак за його застосування у стандартному варіанті

в процесі кластеризації окремих пікселів не враховується текстура окремих областей зображення. Ця обставина добре проявляється на зображеннях, у яких потенційні суперпікселі мають приблизно однакові яскравість та кольоровість, але відрізняються з текстури, що продемонстровано на рисунку 2.16 А.

Стандартна реалізація SLIC не здатна розпізнати на таких зображеннях кордону між областями з різною текстурою і ділить зображення приблизно однакові за розміром області що продемонстровано на рисунку 2.16 Б.

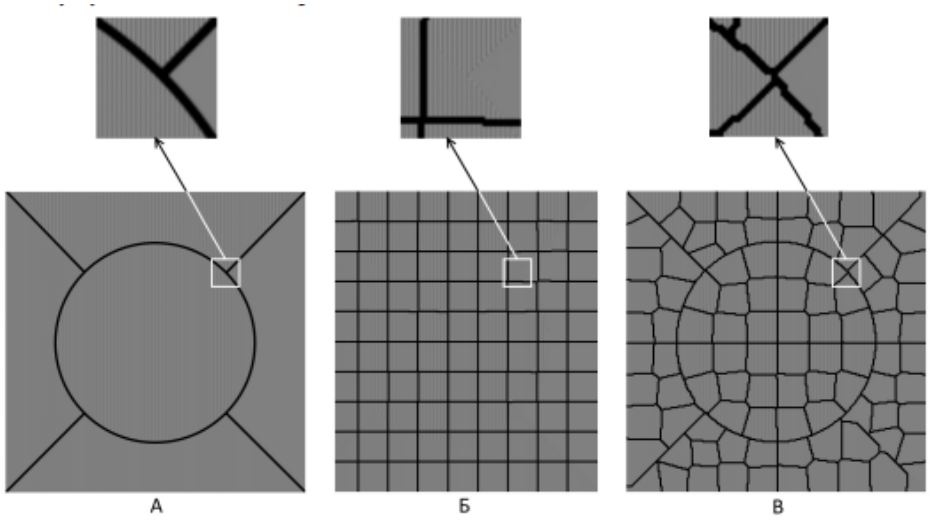


Рисунок 2.16 – Суперпіксельна сегментація зображення з областями приблизно однакового кольору, але з різною текстурою:
А – реальні межі областей, Б – класичний алгоритм SLIC, В – модифікований алгоритм SLIC

Очевидне рішення для цієї проблеми – намагатися розглядати відразу кілька сусідніх пікселів одночасно, наприклад, блоки щоб враховувати взаємозв'язки та текстурні особливості для груп сусідніх пікселів. При цьому, відстанню між деяким блоком і центром кластера буде зважена сума евклідових відстаней між середніми значеннями координат блоку, середніми значеннями колірних компонентів блоку та дисперсіями колірних компонентів блоку.

Область пошуку для кожного кластера блоком розміром 2S × 2S пікселів від- щодо початкового положення центру кластерів. Далі, у циклі, що охоплює такі пункти, виконується ітеративне уточнення результатів сегментації.

Якщо досягнуто необхідна точність, наприклад, різниця центрів кластерів на сусідніх кроках менше заданого порогового значення або виконано задане число ітерацій здійснюється зупинка, в іншому випадку - перехід на наступний крок циклу. На рисунку 2.17В видно, що даний алгоритм вже здатний виявляти межі на зображенні, де стандартний алгоритм виявився безсилим.

Для перевірки даної гіпотези необхідно було згенерувати велику кількість подібних зображень. Це можна зробити на основі тестового набору Берклі, який включає не тільки зображення, але та вручну розмічені межі об'єктів. При зафарбовуванні кожного об'єкта випадково згенерованою текстурою виходять зображення необхідні проведення тесту. Результат роботи модифікованого алгоритму SLIC знову виявляється краще в порівнянні з класичним алгоритмом.

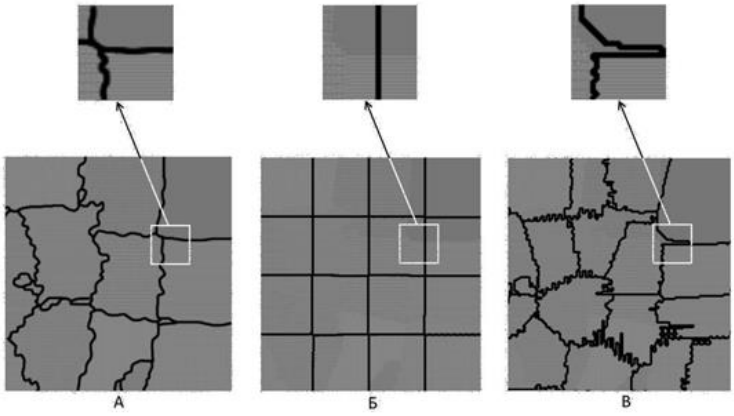


Рисунок 2.17 – Суперпіксельна сегментація фрагмента зображення з областями приблизно однакового кольору, але з різною текстурою, згенерованого використанням тестового набору Берклі :
А – реальні межі областей, Б – класичний алгоритм SLIC, В – модифікований алгоритм SLIC

Порівняння на більшій кількості подібних зображень, підтверджує, що модифікований алгоритм краще виділяє межі об'єктів на таких зображеннях (до 20% більше), дещо поступаючись у надмірності сегментації, що пояснюється більш звивистими межами суперпікселів. Відповідні залежності наведено на рисунку 2.18.

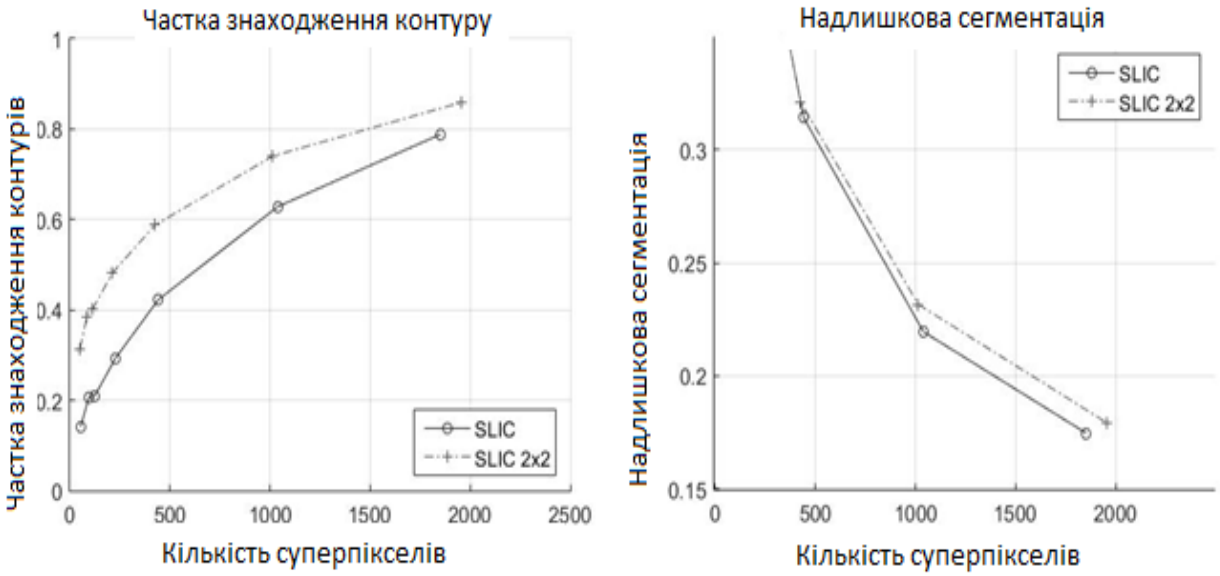


Рисунок 2.18 – Порівняння модифікованого та класичного алгоритмів SLIC на зображеннях з областями, але з різною текстурою (ліворуч – відсоток виявлених меж, праворуч – надмірність сегментації)

При використанні модифікованого алгоритму на довільних зображеннях він також показує непогані результати. Так, при тестуванні на «звичайних» зображеннях з тестового набору Берклі, він виявляє приблизно таку ж кількість кордонів як класичний алгоритм, дещо поступаючись у надмірності сегментації, що показано рисунку 2.19.

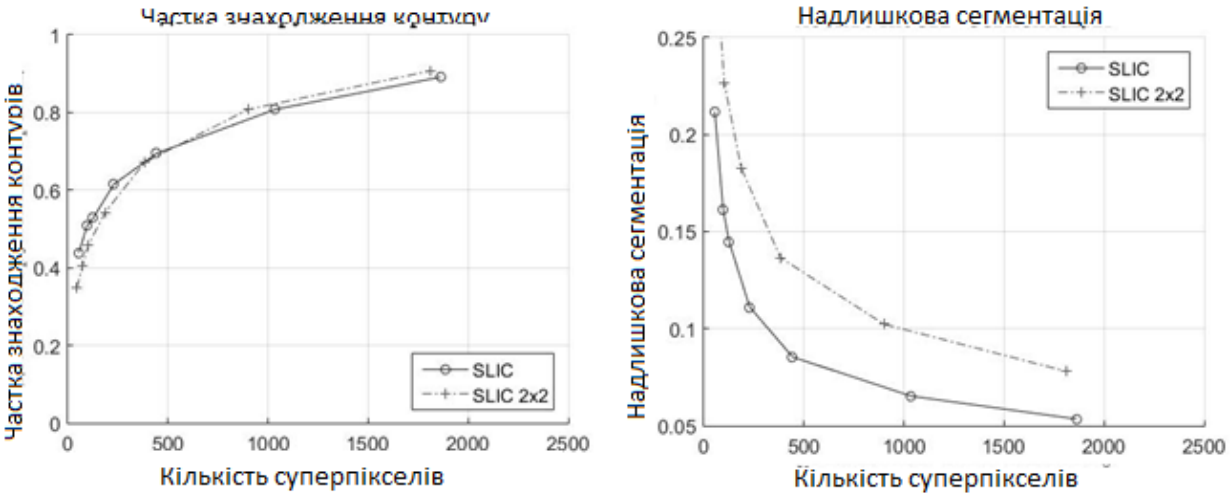


Рисунок 2.19 – Порівняння модифікованого та класичного алгоритмів SLIC на зображеннях з тестового набору Берклі (ліворуч – відсоток виявлених меж, праворуч – надмірність сегментації)

Таким чином, модифікований алгоритм SLIC перевершує в точності оригінальний алгоритм на тих зображеннях, де оригінальний SLIC практично не здатний виділяти межі між суперпікселями, але в середньому має дещо гірші значення відсотка виявлених кордонів і помилок сегментації.

3 КОМП’ЮТЕРНА МОДЕЛЬ СЕГМЕНТАЦІЇ ЗОБРАЖЕНЬ

3.1 Обґрунтування вибору середовища програмної реалізації

У рамках кваліфікаційної роботи був розроблений алгоритм для суперпіксельної сегментації зображень. Для реалізації було обране середовище Python IDLE. Це обумовлено тим, що Python IDLE засновано високопродуктивною мовою Python.

Операційна система Windows надає розроблювачам додатків потужні засоби Інтерфейсу Графічних Пристроїв GDI (Graphics Device Interface) для побудови графічних зображень незалежно від типу використовуваного пристрою висновку. На жаль, GDI обтяжує програмістів безліччю додаткових дій (зокрема, по керуванню системними ресурсами), які відволікають розроблювача від його основного завдання – створення графіки.

Python – високорівнева мова програмування загального призначення з динамічною строгою типізацією та автоматичним управлінням пам'яттю, орієнтована підвищення продуктивності розробника, читання коду та його якості, і навіть забезпечення переносимості написаних у ньому программ.

Мова є повністю об'єктно-орієнтованою – все є об'єктами. Незвичайною особливістю мови є виділення блоків коду пробільними відступами. Синтаксис ядра мови мінімалістичний, рахунок чого практично рідко виникає необхідність звертатися до документации. Сама ж мова відома як інтерпретована і використовується в тому числі для написання скриптів. Недоліками мови є найчастіше нижча швидкість роботи і більш високе споживання пам'яті написаних нею програм порівняно з аналогічним кодом, написаним компілюваними мовами, таких як C або C++.

Python є мультипарадигмальною мовою програмування, що підтримує імперативне, процедурне, структурне, об'єктно-орієнтоване програмування, метапрограмування та функціональне програмування. Завдання узагальненого

програмування вирішуються рахунок динамічної типизации. Аспектно-орієнтоване програмування частково підтримується через декоратори, повноцінна підтримка забезпечується додатковими фреймворками.

Такі методики як контрактне та логічне програмування можна реалізувати за допомогою бібліотек чи розширень. Основні архітектурні риси – динамічна типізація, автоматичне управління пам'яттю, повна інтроспекція, механізм обробки винятків, підтримка багатопоточних обчислень з глобальним блокуванням інтерпретатора, високорівневі структури даних. Підтримується розбиття програм на модулі, які можуть об'єднуватися в пакети, також приклад программи на мові Python вказано на рисунку 3.1.

```python
class Money:

    currency_rates = {
        '$': 1,
        '€': 0.88,
    }

    def __init__(self, symbol, amount):
        self.symbol = symbol
        self.amount = amount

    def __repr__(self):
        return '%s%.2f' % (self.symbol, self.amount)

    def convert(self, other):
        """ Convert other amount to our currency """
        new_amount = (
            other.amount / self.currency_rates[other.symbol]
            * self.currency_rates[self.symbol])

        return Money(self.symbol, new_amount)
```

Рисунок 3.1 – Код на мові Python

Python є мультипарадигмальною мовою програмування, що підтримує імперативне, процедурне, структурне, об'єктно-орієнтоване програмування, метапрограмування та функціональне програмування. Завдання узагальненого програмування вирішуються за рахунок динамічної типізації. Аспектно-орієнтоване програмування частично підтримується через декоратори, повноцінна підтримка забезпечується додатковими фреймворками.

Такі методики як контрактне та логічне програмування можна реалізувати за допомогою бібліотек чи розширень. Основні архітектурні риси – динамічна типізація, автоматичне управління пам'яттю, повна інтроспекція, механізм обробки винятків, підтримка багатопоточних обчислень із глобальним блокуванням інтерпретатора, високорівневі структури даних. Підтримується розбиття програм на модулі, які можуть об'єднуватися в пакети.

Python підтримує динамічну типізацію, тобто тип змінної визначається лише під час виконання. Тому замість "привласнення значення змінної" краще говорити про "зв'язування значення з деяким ім'ям". До примітивних типів у Python відносяться булевий, ціле число довільної точності, число з плаваючою комою та комплексне число. З контейнерних типів у Python вбудовані: рядок, список, кортеж, словник та безліч. Усі значення є об'єктами, зокрема функції, методи, модулі, класи. Додати новий тип можна або написавши клас (class), або визначивши новий тип у модулі розширення (наприклад, написаному мовою C). Система класів підтримує успадкування (одиночне та множинне) та метапрограмування. Можливе успадкування від більшості вбудованих типів та типів розширень.

Дизайн мови Python побудований навколо об'єктно-орієнтованої моделі програмування. Реалізація ООП у Python є добре продуманою, але водночас досить специфічною порівняно з іншими об'єктно-орієнтованими мовами. У мові все є об'єктами – або екземплярами класів, або екземплярами метакласів. Винятком є вбудований базовий метаклас type. Отже, класи насправді є екземплярами метакласів, а похідні метакласи є екземплярами метакласу type.

Метакласи є частиною концепції метапрограмування і надають можливість управління спадкуванням класів, що дозволяє створювати абстрактні класи, реєструвати класи або додавати до них будь-який програмний інтерфейс у межах бібліотеки чи фреймворку. Класи за своєю суттю представляють план або опис того, як створити об'єкт, і зберігають опис

атрибутів об'єкта і методів для роботи з ним. Схему інтерпритатора Python вказано на рисунку 3.2.

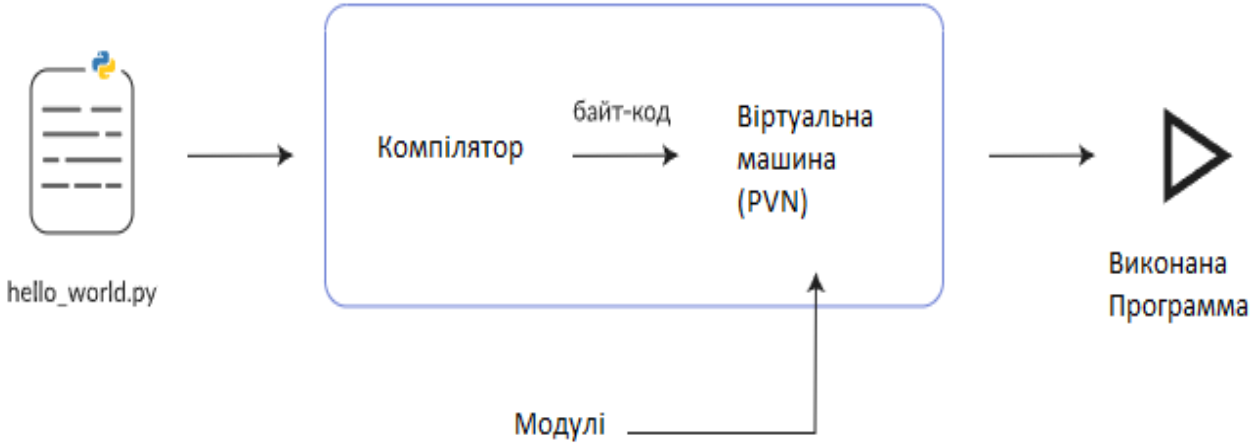


Рисунок 3.2 – Інтерпритатор Python

Парадигма ООП ґрунтується на інкапсуляції, наслідуванні та поліморфізмі. Інкапсуляція в Python представлена можливістю зберігання публічних та прихованих атрибутів (полів) в об'єкті з наданням методів для роботи з ними, при цьому насправді всі атрибути є публічними, але для позначення прихованих атрибутів існує угода про іменування.

Спадкування дозволяє створювати похідні об'єкти без необхідності повторного написання коду, а поліморфізм полягає у можливості перевизначення будь-яких методів об'єкта (у Python усі методи є віртуальними), а також у навантаженні методів та операторів. Перевантаження методів у Python реалізується рахунок можливості виклику однієї й тієї методу з різним набором аргументов. Особливістю Python є можливість модифікувати класи після їх оголошення, додаючи до них нові атрибути та методи, також можна модифікувати й самі об'єкти, у результаті класи можуть використовуватися як структури для зберігання довільних даних.

3.2 Open-CV

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. 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 – ядро містить базові структури даних та алгоритми:

– базові операції над багатовимірними числовими масивами;

– матрична алгебра, математичні ф-ції, 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.)рус., 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 осіб за 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.

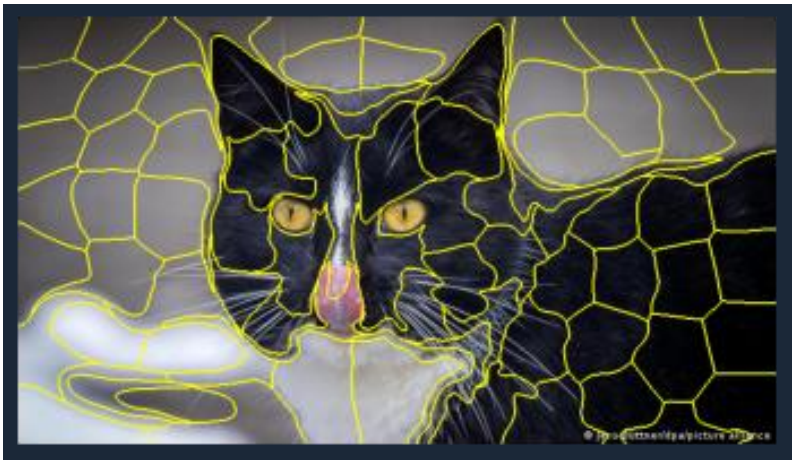
3.3 Програмна реалізація

Для програмної реалізації вибрано дослідження алгоритму SLIC, мову програмування Python з додатком OpenCV. Приклад программи показано на рисунку 3.3.

```
# import the necessary packages
from skimage.segmentation import slic
from skimage.segmentation import mark_boundaries
from skimage.util import img_as_float
from skimage import io
import matplotlib.pyplot as plt



# load the image and convert it to a floating point data type
image = img_as_float(io.imread("image.jpg"))

# loop over the number of segments
for numSegments in (100, 200, 300):
    # apply SLIC and extract (approximately) the supplied number
    # of segments
    segments = slic(image, n_segments = numSegments, sigma = 5)

    # show the output of SLIC
    fig = plt.figure("Superpixels -- %d segments" % (numSegments))
    ax = fig.add_subplot(1, 1, 1)
    ax.imshow(mark_boundaries(image, segments))
    plt.axis("off")

# show the plots
plt.show()
```

Рисунок 3.3 – Реалізація алгоритму SLIC

Алгоритм може розбивати зображення на різну кількість сегментацій як вказано на рисунку 3.4.



(а)

(б)

Рисунок 3.4 – Робота програми:
(а) сегментація на 50 сегментів; (б) сегментація на 100 сегментів

3.4 Інструкція користувача

Для використання програми користувачу потрібно спочатку встановити Python з офіційного сайту, який зображено на рисунку 3.5. Завантажуємо потрібну версію та запускаємо .exe файл. Не забуваймо встановити галочку add path.

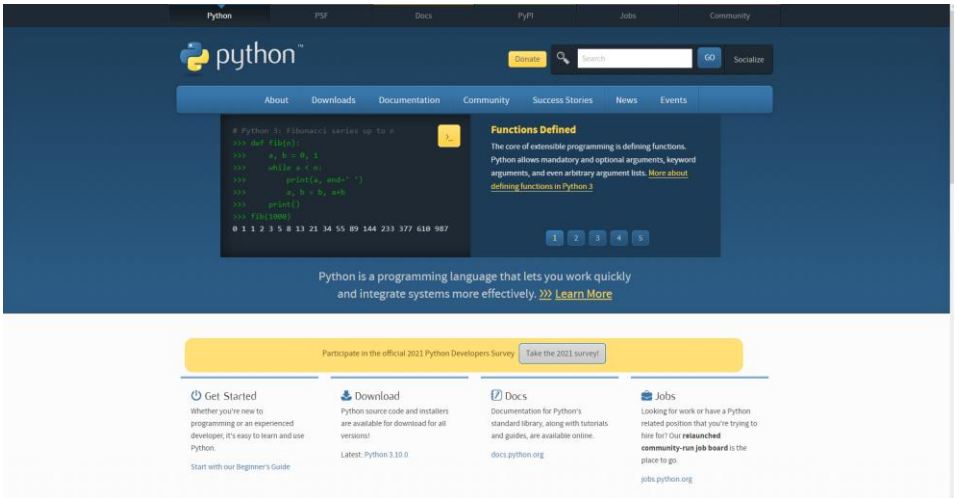


Рисунок 3.5 – Офіційний сайт Python

Встановлюємо virtualenv. Віртуальне середовище нам потрібне для того, щоб для кожного окремого проекту була своя "кімната" зі своїми версіями встановлених бібліотек, які не залежатимуть від інших проектів і плутатимуться між собою. Пакети встановлюватимемо за допомогою pip.

Зазвичай потрібно його оновити командою: python -m pip install --upgrade pip Обновили pip, тепер встановимо віртуальне середовище: pip install virtualenv Командою cd перейдіть до папки, де хочете створити середовище і введіть команду: mkdir opencvtutorial_env — ми створили середовище з назвою opencvtutorial_env. Далі вводимо команду virtualenv opencvtutorial_env і для активації перейдіть до папки середовища і далі за допомогою Tab до activate. .\opencvtutorial_env\Scripts\activate.

Встановимо бібліотеки OpenCV-Python, Numpy та Matplotlib з сайту OpenCV, який продемострировано на рисунку 3.6, які знадобляться для тестування функцій opencv. Вставимо pip install jupyterlab та запустимо його командою jupyter notebook.

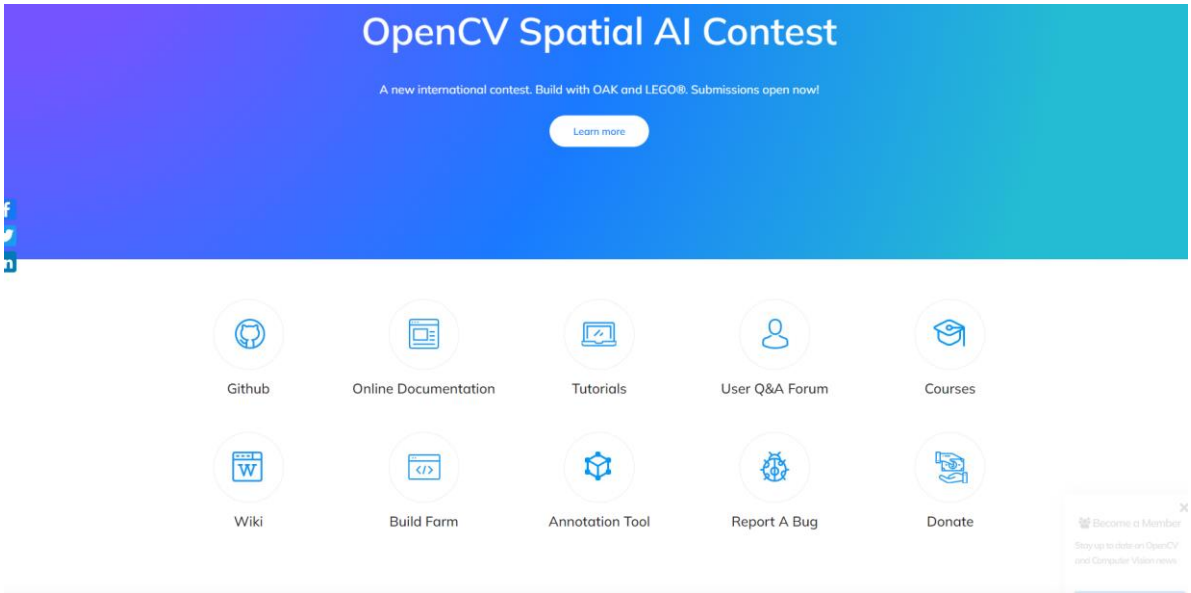


Рисунок 3.6 Домашня сторінка сайту OpenCV

Тепер залишилося перевірити, чи все у нас працює. У вікні, що відкрилося відкриваемо наш файл та користуємося программою

В проекті можливо змінювати кількість сегментів на будь яку ви хочете так як вказано на рисунку 3.7.

```
# import the necessary packages
from skimage.segmentation import slic
from skimage.segmentation import mark_boundaries
from skimage.util import img_as_float
from skimage import io
import matplotlib.pyplot as plt


# load the image and convert it to a floating point data type
image = img_as_float(io.imread("image.jpg"))

# loop over the number of segments
for numSegments in (100, 200, 300):
    # apply SLIC and extract (approximately) the supplied number
    # of segments
    segments = slic(image, n_segments = numSegments, sigma = 5)

    # show the output of SLIC
    fig = plt.figure("Superpixels -- %d segments" % (numSegments))
    ax = fig.add_subplot(1, 1, 1)
    ax.imshow(mark_boundaries(image, segments))
    plt.axis("off")

# show the plots
plt.show()
```

Рисунок 3.7 – Зміни кількості сегментів

3.5 Тестування розробленої моделі

Для тестування программи потрібно перевірити основний функціонал программи а саме сегментацію.Пі рис 3.8.

Для того щоб розпочати тестування потрібно запустите проект та спостерігати як змінюється вихідне зображення.

Під час тесту було отримано сегментації, що продемонстровані на рисунку 3.8.

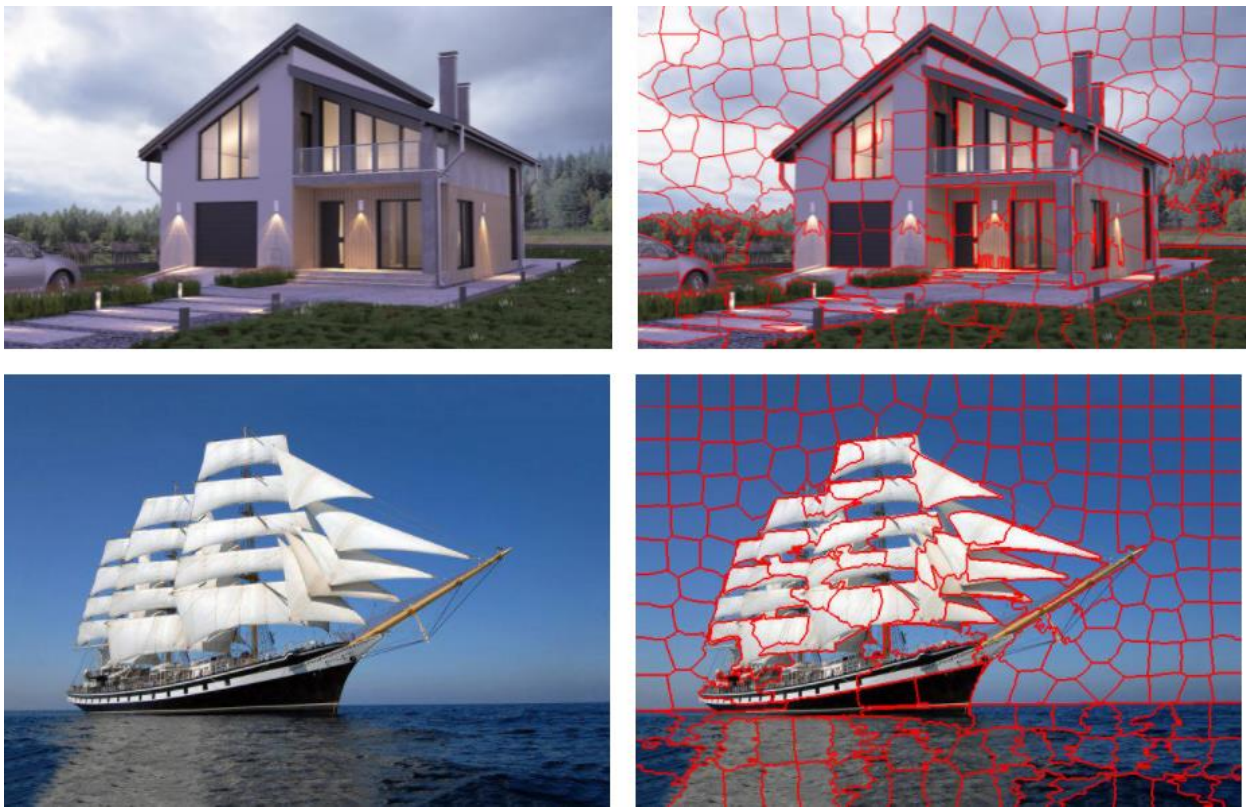
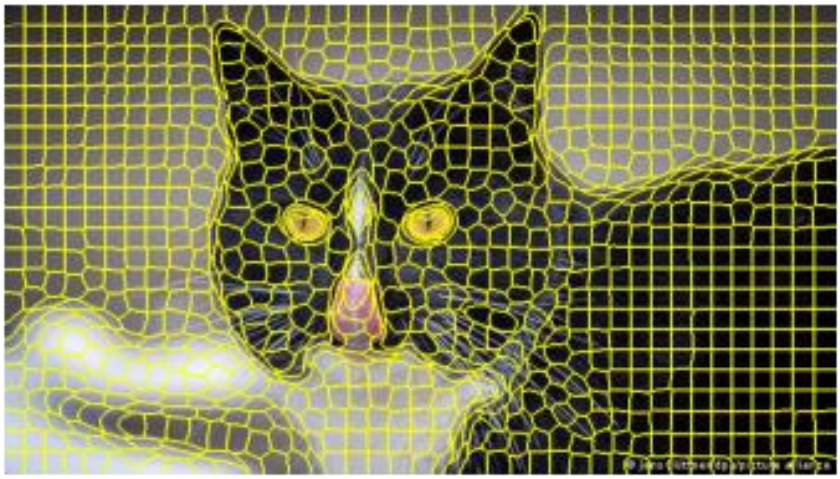


Рисунок 3.8 – Різні приклади роботи програми

Також потрібно перевірити сегментацію на різній кількості сегментів для одного зображення, наприклад для 50 сегментів (рисунок 3.9(а)) та 1000 сегментів (рисунок 3.9(б)).

(а) Для 50



(б) Для 1000

Рисунок 3.9 (а-б) – Робота для 50 та 1000 сегментів

4 ВИСНОВКИ

У рамках кваліфікаційної роботи був розроблений і реалізований метод сегментаціїї зображень.

У роботі проведено порівняння основних підходів до побудови алгоритмів сегментації, виміряно відносні показники швидкості та точності для різних архітектур алгоритмів. Отримані експериментальні дані говорять про тому, що найточніші результати досягаються при використанні пересегментації методом SLIC, використання як статистичних, так і текстурних ознак суперпікселів. Серед методів машинного навчання кращі результати за точністю класифікації дає випадковий ліс (RF).

Було сформульовано загальні вимоги до базових ознак, що швидко формуються при первинній сегментації. На підставі аналізу ряду типових завдань обробки та розпізнавання зображень введена в розгляд система, що складається з базових ознак, які в першу чергу слід включати в суперпікселі. Показано, як ці базові ознаки перераховуються у похідні ознаки, що безпосередньо використовуються при вирішенні конкретних прикладних завдань.

Досяжна точність семантичної сегментації становить близько 88,7%. З використанням контекстної інформації (через модель CRF) можна досягти точності близько 89,8%. Для підвищення швидкості роботи необхідно відмовитися від розбиття зображення на суперпікселі в користь розбиття на квадратні осередки, використовувати лише статистичні ознаки зображень та не використовувати CRF. Крім того, можна знизити кількість вирішальних дерев у класифікаторі без суттєвого зниження його точності.

5 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 with Fusion 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. *British Journal of Mathematical and Statistical Psychology*, *59*(1), 1-34.

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

6. Xu, J., Han, J., Xiong, K., & Nie, F. (2016, July). Robust and Sparse Fuzzy K-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). *Introduction to time series analysis and forecasting*. John Wiley & Sons.

15. Zhang, J., Zhao, Z., Xue, Y., Chen, Z., Ma, X., & Zhou, Q. (2017). Time series 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 to Study the Structure of Blood Components.

33. Ren and Malik, "Learning a classification model for segmentation," in Proceedings Ninth IEEE International Conference on Computer Vision, Oct 2003, pp. 10–17 vol.1.

34. M. Miyama, "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), Nov 2017, pp. 278–283.

35. F. Gu, H. Zhang, and C. Wang, "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, July 2018, pp. 6671–6674.

36. R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. S¨usstrunk, "Slic superpixels," 2010.

37. M. R. Luo, CIELAB. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 1–7. [Online]. Available: https://doi.org/10.1007/978-3-642-27851-8 11-1 (Accessed 2020-01-30).

38. R. Achanta and S. Susstrunk, "Superpixels and polygons using simple non-iterative clustering," 07 2017, pp. 4895–4904.

39. M. Van den Bergh, X. Boix, G. Roig, B. Capitani, and L. Van Gool, "SEEDS: Superpixels Extracted via Energy-Driven Sampling," in International Journal of Computer Vision, vol. 111, 10 2012.

40. F. Meyer, "Color image segmentation," in 1992 International Conference on Image Processing and its Applications, April 1992, pp. 303–306.

41. P. Neubert and P. Protzel, “Compact Watershed and Preemptive SLIC: On Improving Tradeoffs of Superpixel Segmentation Algorithms,” in Proceedings - International Conference on Pattern Recognition, 08 2014, pp. 996–1001.

42. P. Neubert and P. Protzel, “Superpixel benchmark and comparison,” in Forum Bildverarbeitung, 2012.

43. D. Martin, C. Fowlkes, D. Tal, and J. Malik, “A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics,” in Proc. 8th Int’l Conf. Computer Vision, vol. 2, July 2001, pp. 416–423.

44. P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, “Contour detection and hierarchical image segmentation,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 33, no. 5, pp. 898–916, May 2011.

45. E. Fykse, «Performance Comparison of GPU, DSP and FPGA implementations of image processing and computer vision algorithms in embedded systems», Norwegian University of Science and Technology, 2013.