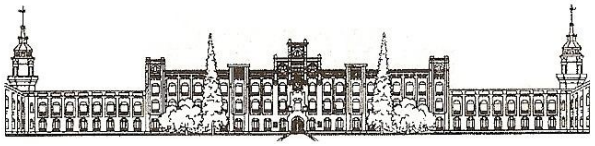


Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут»

Лабораторна робота №3

з дисципліни «Комп'ютерний зір»

«Сегментація зображень»

Виконали студентки групи: КВ-11

ПІБ: Михайліченко Софія Віталіївна
Шевчук Ярослава Олегівна

Перевірила: _____

Київ 2024

Постановка задачі:

Реалізувати наведені нижче завдання у вигляді відповідного застосунку/застосунків, що передбачають завантаження зображень із файлів, їх обробку відповідним чином та візуалізацію і збереження файлів результату (рекомендовані мови програмування: C++ або Python, але за бажання припустимі і інші).

Для роботи із зображеннями рекомендовано використовувати бібліотеку OpenCV (зокрема, cv::Mat, cv::imread, cv::imwrite, cv::imshow, cv::waitKey), хоча, за бажання припустимо використовувати і інші бібліотеки, що дозволяють роботу з зображеннями (завантаження, збереження, візуалізація, доступ до значень пікселів). Завдання мають бути виконані самостійно, не використовуючи існуючі реалізації відповідних алгоритмів (в т.ч. реалізацію операції згортки) з бібліотеки OpenCV або інших бібліотек.

Завдання мають бути виконані самостійно, не використовуючи (особливо що стосується безпосередньо завдань лабораторної роботи) існуючі реалізації відповідних алгоритмів (в т.ч. реалізацію операції згортки) з бібліотеки OpenCV або інших бібліотек. Реалізацію допоміжних алгоритмів (наприклад, виконання згладжування, обчислення частинних похідних, виконання згортки тощо) тбажано взяти із попередніх лабораторних робіт (але в крайньому випадку допускається брати відповідні реалізації з OpenCV). Допоміжні засоби (такі як колекції, загальні алгоритми, математичні функції, комплексні числа, лінійна алгебра тощо) можуть бути як реалізовані самостійно, так і отримані із використанням бібліотек (STL, Eigen тощо).

Додаткові завдання можуть бути виконані для отримання додаткових балів. В звіті потрібно навести, зокрема, код програми, приклади виконання реалізованих процедур (де можливо із різними параметрами), а також короткі висновки.

Завдання:

1. Реалізувати процедуру бінаризації зображення за заданим глобальним порогом. Для деякого півтонового зображення виконати процедуру бінаризації для різних значень порогів (підібрати оптимальний для виділення бажаних об’єктів).

2. Реалізувати визначення порогу бінаризації методом Оцу. Виконати бінаризацію та порівняти результати (бінаризовані зображення та значення порогу) із підібраним в п.1.

3. Реалізувати процедуру стилізації Віннемьоллера. Виконати бінаризацію зображень на основі стилізації Віннемьоллера.

4. Реалізувати процедуру сегментації зображеннь шляхом вирощування насіння. Відношення еквівалентності обрати самостійно (на вибір для півтонових або кольорових зображень; достатньо обрати просте відношення еквівалентності, наприклад, поділити множину значень яскравості на інтервали). Побудувати карту сегментів, позначаючи сегменти кольорами і/або рівнями яскравості.

5. Реалізувати сегментацію зображень за допомогою алгоритму зсуву середнього (наприклад, використовуючи квадратні/кубічні вікна в просторі ознак, що попарно не перетинаються). Для цього побудувати простір ознак (багатовимірна гістограма), та виконати зсув середнього (наприклад, попередньо розбивши простір ознак на вікна, та виконуючи зсув середнього для кожного вікна, об’єднуючи їх у випадку співпадіння середніх), після чого об’єднати ознаки, відповідні фінальні середні для яких є близькими у класи еквівалентності. Після цього для отримання безпосередньо сегментації може бути застосований алгоритм вирощування насіння (із отриманими класами еквівалентності). В якості ознак можна скористатися локальним середнім та стандартним відхиленням (у випадку півтонових зображень), або і безпосередньо значеннями каналів (для кольорового зображення).

6. Реалізувати сегментацію зображень за допомогою алгоритму поширення довіри (член даних, параметр члена гладкості Поттса та критерій зупинки обрати самостійно).

Додаткові завдання (за бажанням):

1. Порівняти реалізовані алгоритми із наявними в OpenCV (де це можливо).
2. Дослідити вплив позицій випадкових початкових точок у алгоритмі сегментації зображень шляхом вирощування насіння, якщо відношення еквівалентності не застосовується.(завдання 4)
3. Дослідити вплив членів даних, членів гладкості, критеріїв зупинки при сегментації зображень за допомогою алгоритму поширення довіри.(завдання 6)

Порядок виконання роботи

Завдання 1

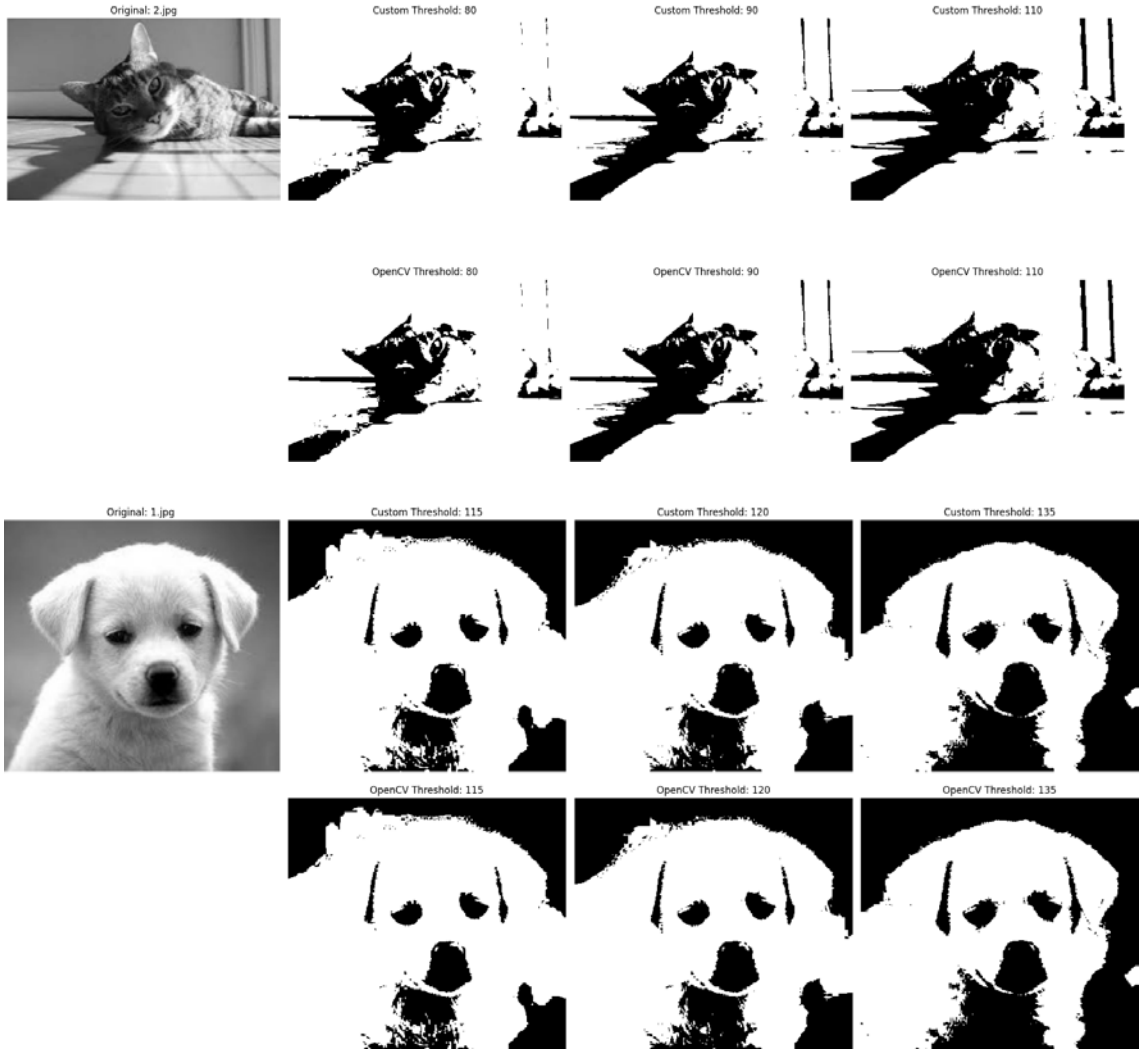
Реалізувати процедуру бінаризації зображення за заданим глобальним порогом. Для деякого півтонового зображення виконати процедуру бінаризації для різних значень порогів (підібрати оптимальний для виділення бажаних об’єктів).

Бінаризація — це процес перетворення зображення в півтонах (градаціях сірого) у двійкове (чорно-біле), де кожен піксель набуває значення 0 або 255 залежно від порогу, що визначає межу між світлими і темними областями. Ця процедура є основою багатьох алгоритмів комп’ютерного зору, оскільки вона спрощує аналіз зображення, зменшуючи його складність і виділяючи ключові об’єкти або контури.

Глобальний поріг — це фіксоване значення інтенсивності, яке застосовується до всього зображення. Якщо інтенсивність пікселя більша за поріг, він стає білим (255), в іншому випадку — чорним (0).

У цьому завданні реалізовано процедуру бінаризації зображення на основі глобального порогу. Процес включає застосування декількох порогів для кожного зображення, що дозволяє підібрати оптимальні параметри для виділення бажаних об’єктів.

Результати виконання:



Порівняння з функціями OpenCV:

OpenCV використовує функцію cv2.threshold, яка оптимізована для швидкості та ефективності. Вона також надає додаткові параметри, такі як різні методи бінаризації (наприклад, cv2.THRESH_BINARY_INV, cv2.THRESH_TRUNC). OpenCV зазвичай швидший завдяки використанню оптимізованих алгоритмів. OpenCV підтримує адаптивну бінаризацію через функцію cv2.adaptiveThreshold. Ця функція дозволяє змінювати поріг для різних ділянок зображення, що особливо корисно для зображень з неоднорідним освітленням.

Самостійно реалізований метод, заснований на NumPy, хоча й простий, може бути повільнішим через накладні витрати на обробку масивів. Хоча як бачимо, результати є подібними, що також свідчить про коректну реалізацію.

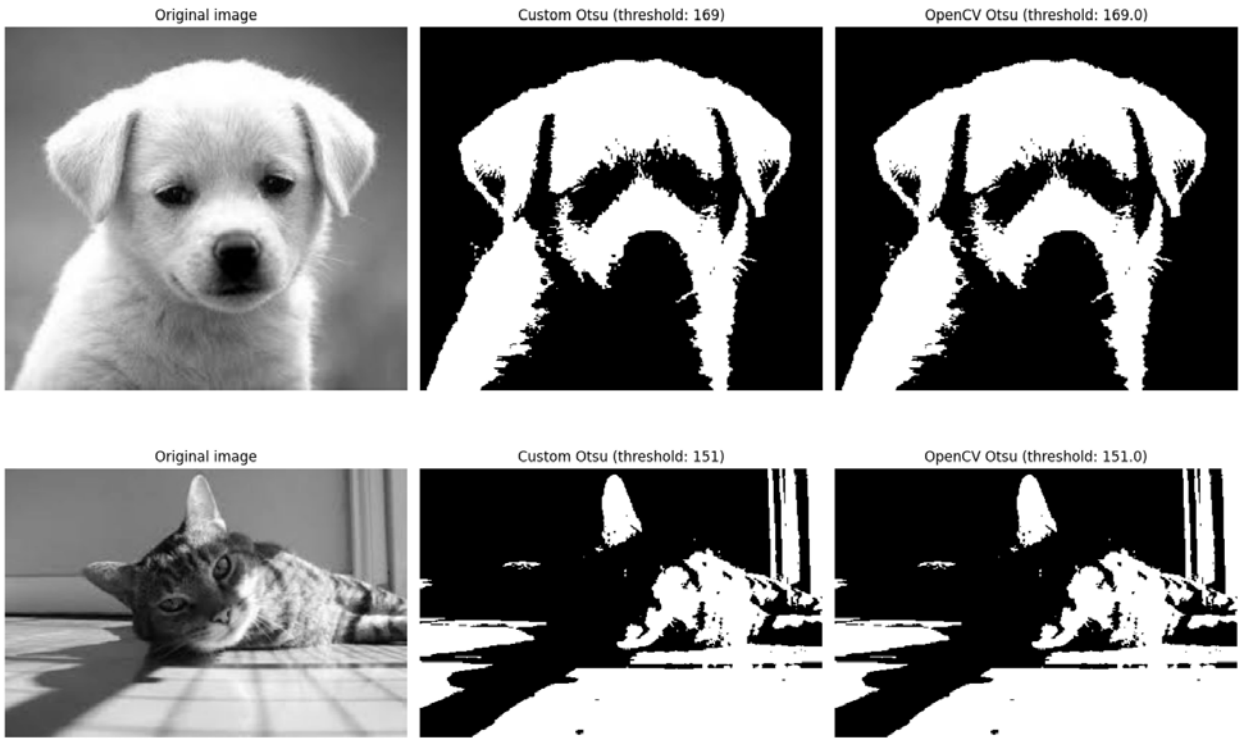
Завдання 2

Реалізувати визначення порогу бінаризації методом Оцу. Виконати бінаризацію та порівняти результати (бінаризовані зображення та значення порогу) із підібраним в п.1.

Завдання полягало у визначенні оптимального порогу бінаризації зображення за допомогою методу Оцу. Реалізація виконувалася двома способами: через власну функцію та за допомогою бібліотеки OpenCV. Метод Оцу дозволяє автоматично обирати такий поріг яскравості, який найкраще розділяє зображення на два класи — фон і об’єкт. Для кожного зображення були отримані результати бінаризації, а також обчислені значення порогу. В процесі порівняли результати, отримані власною функцією, з результатами OpenCV. Крім того, досліджували вплив різних фіксованих порогів на вигляд бінаризованого зображення. Світліші результати бінаризації вказують на застосування нижчого порогу, тоді як вищий поріг залишає більше пікселів чорними. Це дало змогу краще зрозуміти, як змінюється результат залежно від обраного значення.

Метод Оцу, автоматично обираючи оптимальний поріг, демонструє перевагу у випадках, коли необхідно швидко визначити оптимальне значення без ручного тестування. Глобальна бінаризація є корисною, якщо відомі конкретні вимоги до обробки зображення.

Результати виконання:



Порівняння з функціями OpenCV:

Порівняння реалізованого вручну методу Оцу з OpenCV демонструє схожість результатів як за обраним порогом, так і за якістю бінаризації. Реалізація вручну дає змогу краще зрозуміти логіку обчислення: визначення ваги фону та об'єкта, розрахунок міжкласової дисперсії та вибір оптимального порога, що максимізує цю дисперсію. Цей підхід наочно ілюструє, як формується поріг для сегментації.

Водночас реалізація в OpenCV є більш оптимізованою та значно швидшою. Вона використовує ті самі принципи, але реалізована у вигляді високопродуктивного алгоритму, що забезпечує точність і швидкість обчислень, зменшуючи ймовірність помилок у коді. OpenCV автоматично генерує той самий поріг, що й вручну, показуючи відповідність методик. Обидва підходи дозволяють отримати бінаризоване зображення, де об'єкти чітко виділяються на фоні, але OpenCV є більш практичним для реальних проєктів, завдяки своїй простоті та продуктивності.

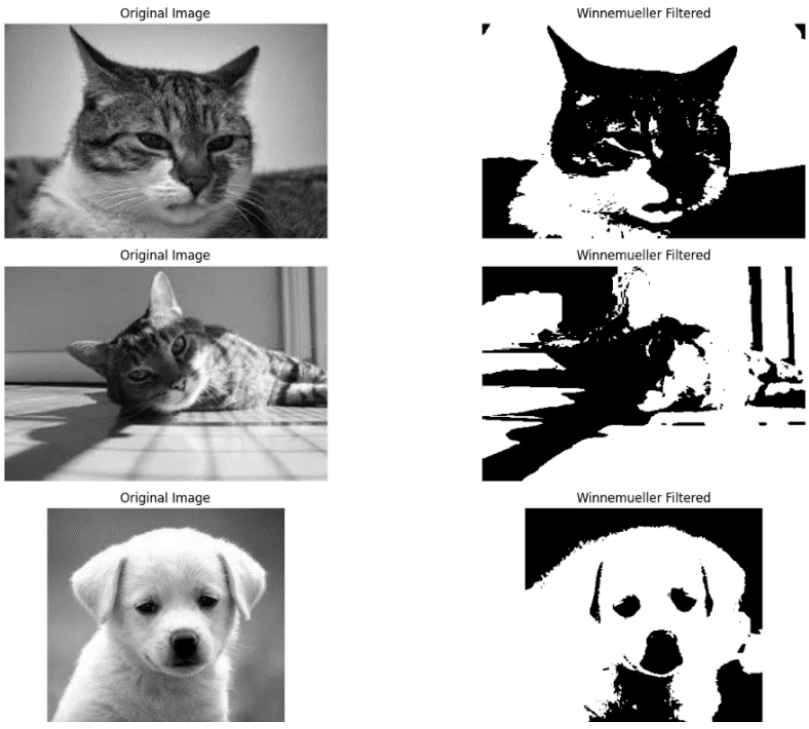
Завдання 3

Реалізувати процедуру стилізації Віннемьоллера. Виконати бінаризацію зображень на основі стилізації Віннемьоллера.

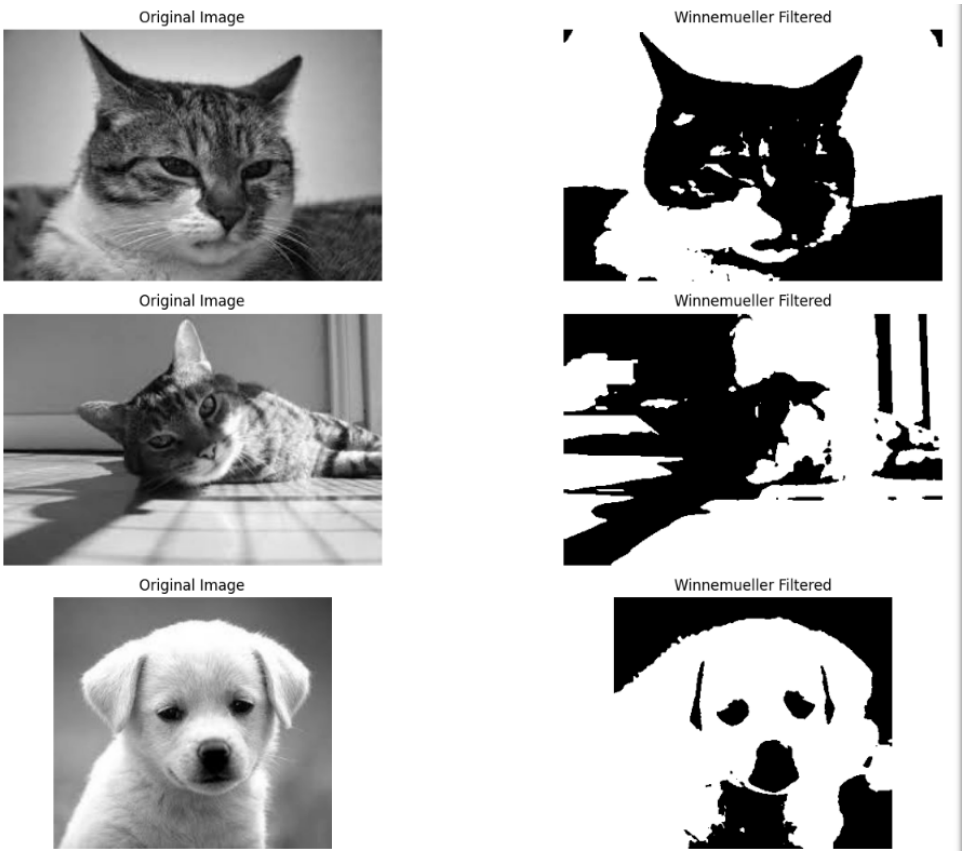
Процедура стилізації Віннемьоллера є потужним методом для перетворення зображень, який дозволяє досягти унікального художнього ефекту. У нашому випадку реалізація цього методу включає кілька етапів, таких як розмиття, виявлення контурів та бінаризація. Бінаризація, виконана на основі стилізації, дозволяє виділити об'єкти на зображенні, перетворюючи їх у чорно-білий формат. Стилізація Віннемьоллера значно збільшує контрастність зображення, підкреслюючи деталі, що можуть бути непомітними в оригіналі. Високий контраст і виразні контури роблять бінаризовані зображення більш чіткими і виразними. Але також Результати сильно залежать від вибору параметрів, таких як sigma, alpha, tau, epsilon, phi, та порогу для бінаризації. Це потребує ручного налаштування для отримання оптимальних результатів.

Результати виконання:

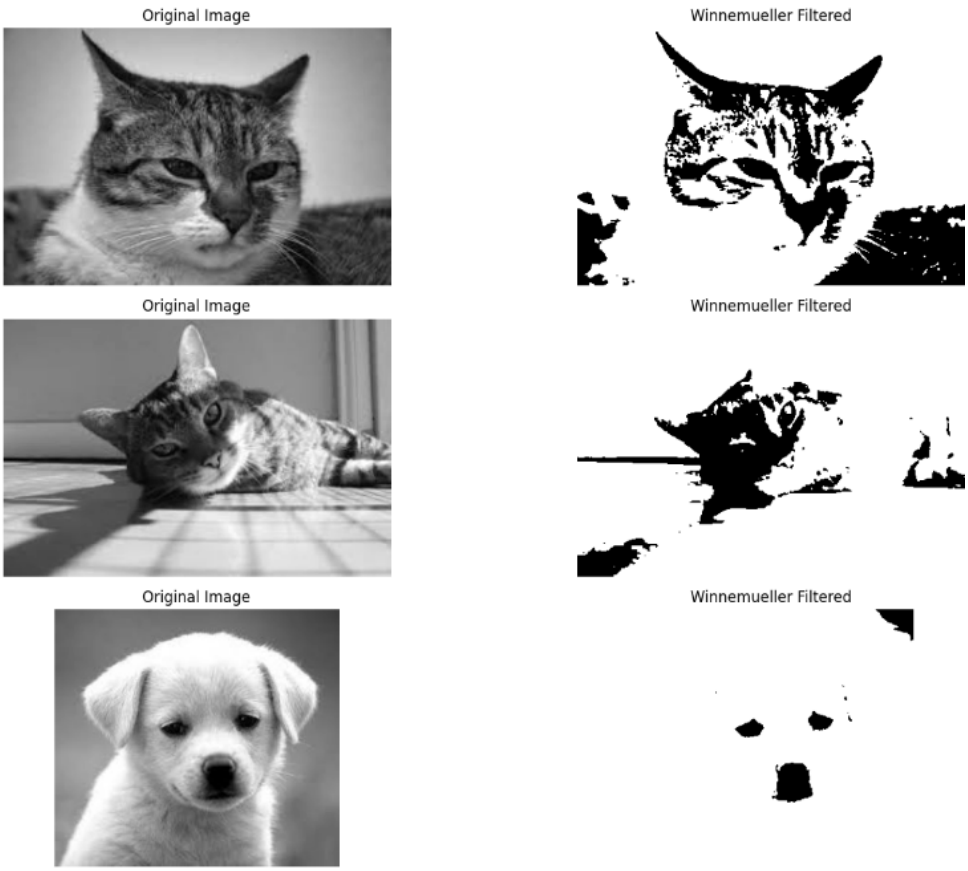
sigma=0.5, alpha=3, tau=0.05, epsilon=0.5, phi=50, threshold=127



sigma=1.5, alpha=3, tau=0.05, epsilon=0.5, phi=50, threshold=127



sigma=0.5, alpha=3, tau=0.01, epsilon=0.3, phi=50, threshold=127



Завдання 4

Реалізувати процедуру сегментації зображенн шляхом вирощування насіння. Відношення еквівалентності обрати самостійно (на вибір для півтонових або кольорових зображень; достатньо обрати просте відношення еквівалентності, наприклад, поділити множину значень яскравості на інтервали). Побудувати карту сегментів, позначаючи сегменти кольорами і/або рівнями яскравості.

Для реалізації процедури сегментації зображень за допомогою методу вирощування насіння ми використовуємо принцип, при якому сегмент формується шляхом додавання сусідніх пікселів до початкового насіння, якщо вони задовольняють визначену умову подібності. У цьому випадку подібність визначається як різниця в яскравості, що не перевищує заданий поріг.

Алгоритм розроблений у вигляді двох підходів: власна реалізація та використання вбудованих функцій OpenCV. Спочатку читається зображення, яке обов'язково має бути в градаціях сірого. Перший підхід базується на ітеративному аналізі сусідніх пікселів за допомогою стеку, а другий використовує функцію cv2.floodFill, яка реалізує схожу ідею з більшою оптимізацією.

У власній реалізації створюється масив, що позначає вже оброблені пікселі, а також сегментоване зображення. Починається з точки насіння, яка додається до стеку. Далі в циклі перевіряються всі сусідні пікселі на відповідність порогу різниці яскравості. Якщо піксель відповідає умовам, він додається до стеку для подальшої обробки, і його координати позначаються як відвідані.

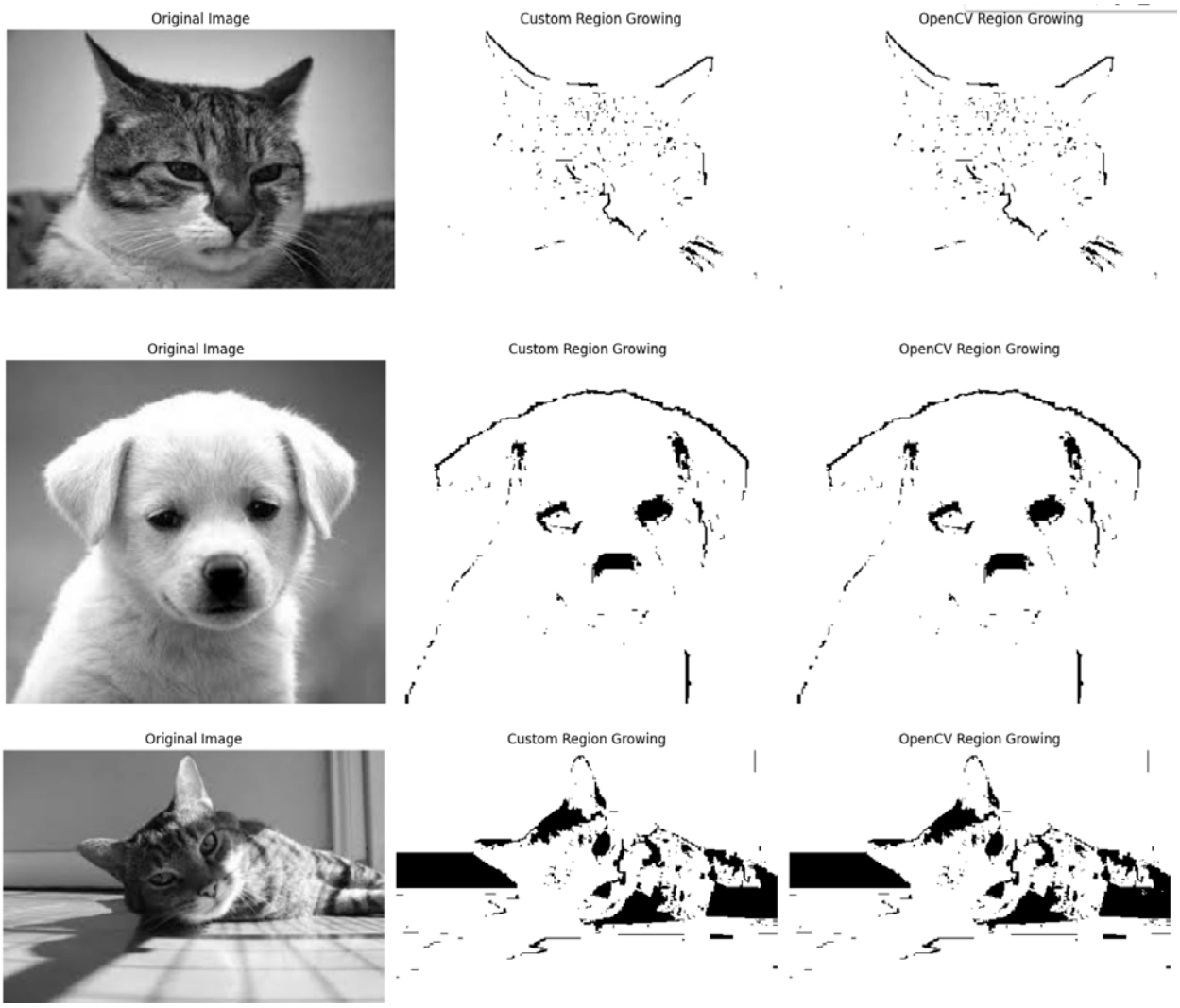
Використовуючи OpenCV, функція cv2.floodFill автоматизує цей процес, працюючи з маскою, яка контролює зони сегментації. Тут задається початкове значення пікселя та межі подібності, визначені порогом.

Для візуалізації створюється порівняння результатів обох підходів. На одному зображенні демонструється оригінал, а на інших – результати сегментації,

отримані за допомогою кожного з методів. Для цього використовується бібліотека Matplotlib.

У коді важливо врахувати правильність введення координат точки насіння, коректність порогового значення та формат вхідного зображення. В результаті на виході отримується карта сегментів, де сегменти позначаються кольором (або рівнями яскравості для чорно-білих зображень).

Результати виконання:



Порівняння з функціями OpenCV:

Порівняння власної реалізації вирощування насіння із вбудованими функціями OpenCV дає змогу оцінити ефективність обох підходів для задачі сегментації зображень.

Власна реалізація працює за принципом ручного обходу сусідніх пікселів, додаючи їх до сегмента, якщо вони відповідають заданим критеріям подібності. Цей підхід передбачає використання циклів і стеку для зберігання пікселів, які підлягають перевірці. Вона гнучка і легко модифікується, наприклад, для використання інших критеріїв еквівалентності або для врахування складніших умов сегментації. Однак цей метод має меншу оптимізацію і може бути повільнішим для великих зображень через велику кількість ітерацій та ручний контроль за всіма процесами.

Функція cv2.floodFill з бібліотеки OpenCV автоматизує процес вирощування насіння, зменшуючи обсяг коду і прискорюючи обчислення. Вона приймає точку насіння, допустимі межі різниці між значенням інтенсивності пікселів і порогом, а також спеціальну маску, яка обмежує область, де виконується заливка. Завдяки внутрішнім оптимізаціям OpenCV цей підхід є значно швидшим і надійнішим для стандартних задач сегментації. Втім, він менш гнучкий, оскільки дозволяє працювати тільки з визначеними умовами подібності, такими як допустима різниця яскравості.

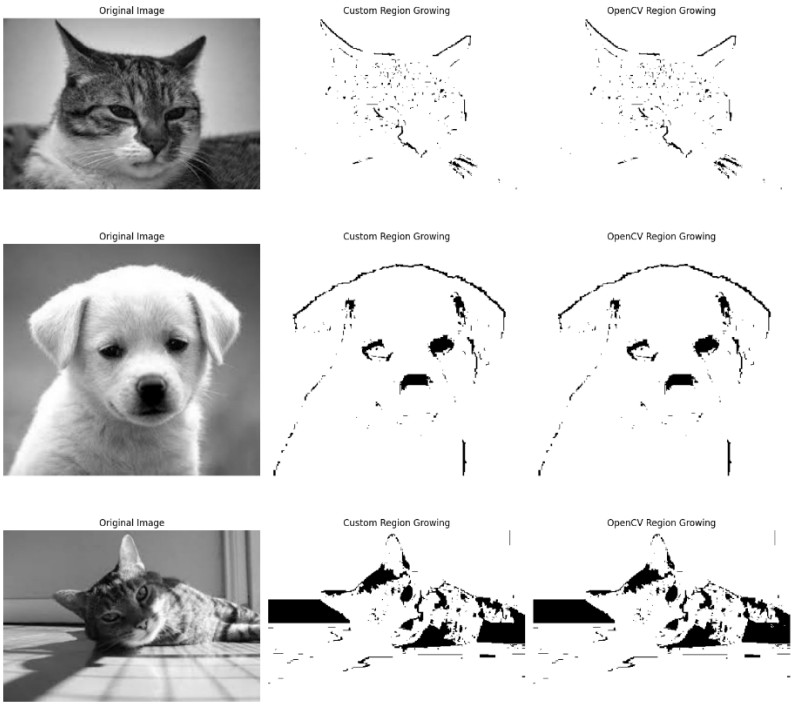
Для оцінки обох підходів використовуються візуальні порівняння. Результати сегментації, створені кожним методом, демонструються поруч, що дозволяє побачити відмінності у точності та ефективності. На простих зображеннях з різкими межами між сегментами результати обох методів зазвичай подібні. Проте для більш складних зображень, де є шум або нечіткі межі, OpenCV може демонструвати кращу продуктивність завдяки оптимізованим алгоритмам, тоді як власний метод може потребувати доопрацювання для отримання аналогічних результатів.

Таким чином, обидва методи мають свої переваги. Власна реалізація добре підходить для навчальних цілей, глибшого розуміння алгоритму і виконання нестандартних завдань. OpenCV забезпечує швидкість і простоту для практичного застосування, особливо коли критерії подібності чітко визначені.

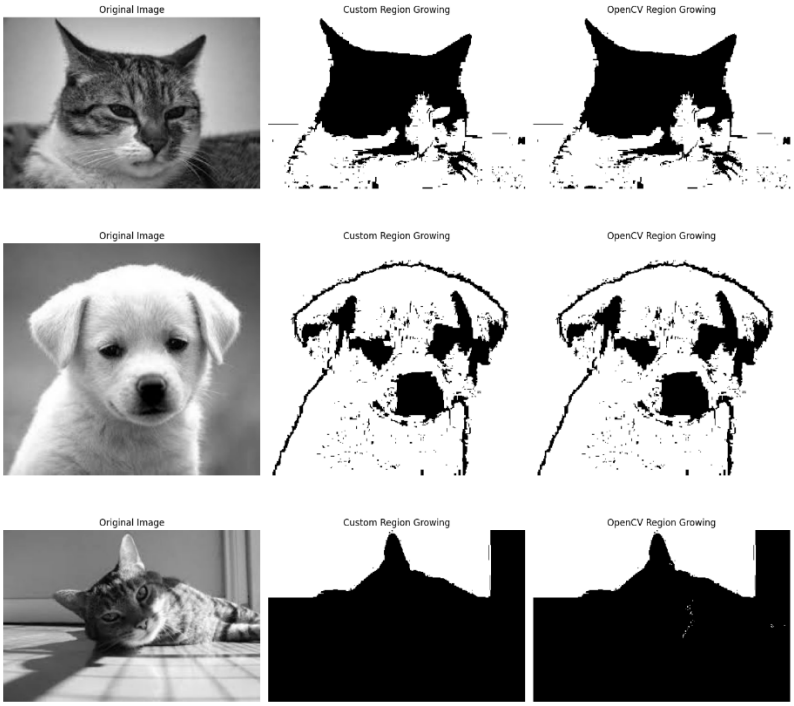
Додаткове завдання:

Дослідити вплив позицій випадкових початкових точок у алгоритмі сегментації зображень шляхом вирощування насіння, якщо відношення еквівалентності не застосовується.

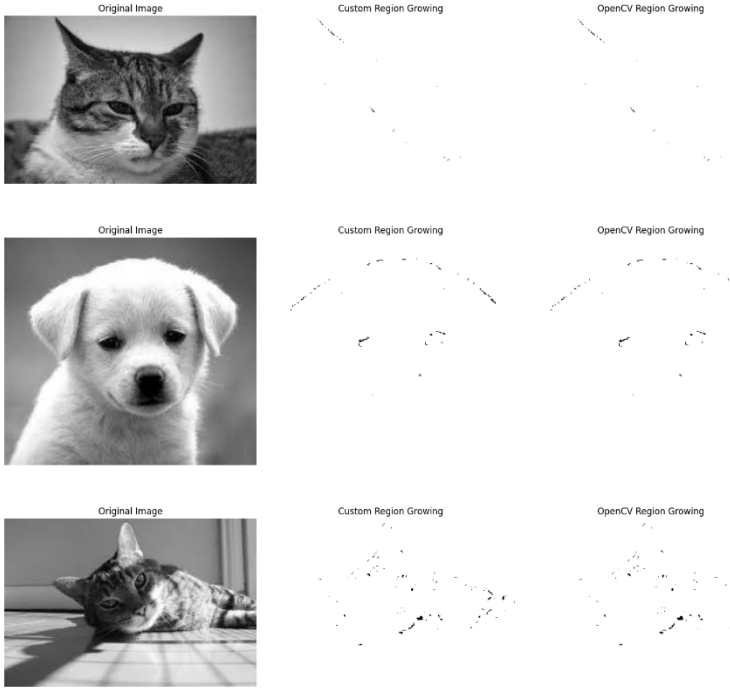
seed_points = (100, 150) threshold = 10



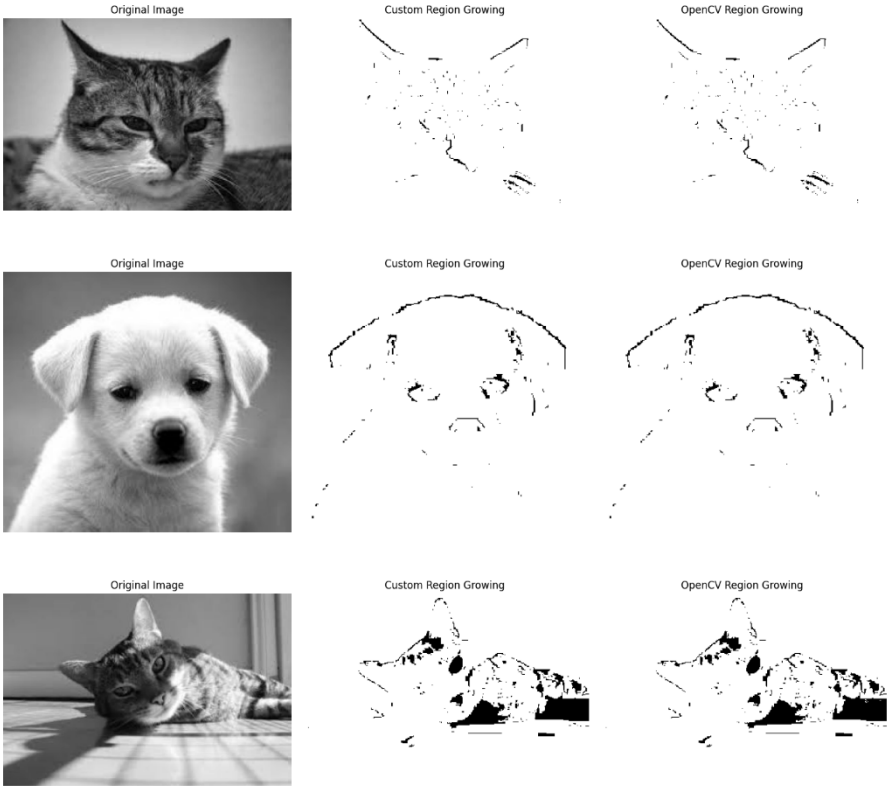
seed_points = (50, 80) threshold = 5



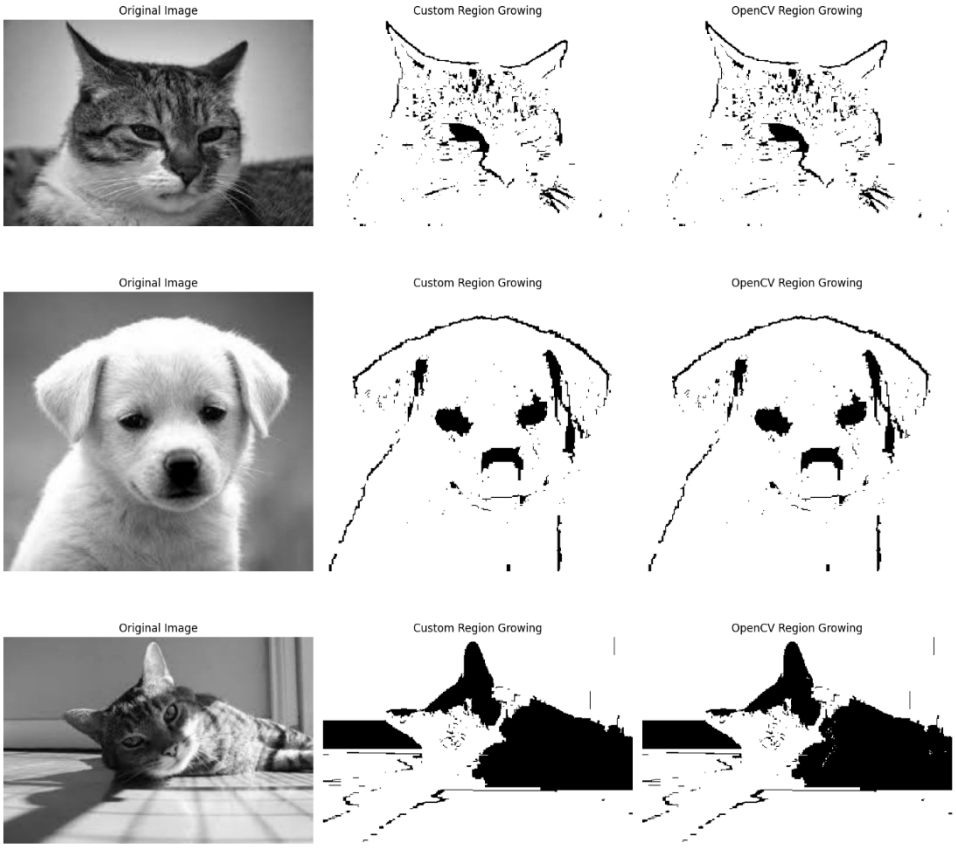
seed_points = (300, 50) threshold = 20



seed_points = (250, 250) threshold = 12



seed_points = (250, 250) threshold = 8



При зміні параметрів seed_points та threshold в алгоритмі вирощування насіння відбувають зміни, які суттєво впливають на результат сегментації зображення. Параметр seed_points визначає початкову точку, з якої алгоритм починає свою роботу. Це важливий момент, оскільки з того місця, де ми вибираємо початкову точку, залежатиме, яку частину зображення буде охоплено сегментацією. Якщо точка насіння розташована на контрастній межі між двома різними ділянками зображення (наприклад, між світлими і темними частинами), алгоритм розділить ці частини, створивши чіткі сегменти. Якщо ж точка знаходиться на однорідному фоні, алгоритм може створити більший сегмент, що охоплює велику частину цього фону. Вибір точок у різних частинах зображення також змінює кількість сегментів. Наприклад, вибір точки в центрі може привести до більшої кількості сегментів в середині зображення, тоді як точка в верхньому лівому або нижньому правому куті може визначити, які частини зображення будуть сегментовані першими.

З іншого боку, параметр threshold визначає, наскільки схожими повинні бути пікселі для того, щоб вони потрапляли в один сегмент. Коли поріг низький, скажімо, 5 або 8, алгоритм буде чутливим до навіть малих відмінностей у кольорі чи яскравості пікселів. Це призводить до того, що сегментація буде дуже деталізованою, а сегменти — дрібними. Таке налаштування корисне для зображень, де важлива висока точність і виражені контрасти між різними частинами зображення. У випадку середнього порога, скажімо, 10 чи 15, алгоритм дозволяє трохи більше варіацій у значеннях пікселів, тому сегменти будуть більшими, і сегментація стане менш детальною, але й більш гнучкою для зображень з помірним контрастом. Вищий поріг, наприклад, 20 або 50, робить алгоритм менш чутливим до відмінностей між пікселями, тому сегментація буде більш загальною, створюючи великі сегменти, де пікселі з незначними відмінностями об'єднуються в один сегмент. Це може бути корисно для зображень з низьким контрастом або якщо потрібно зменшити кількість сегментів.

Таким чином, зміна параметрів seed_points та threshold дозволяє тонко налаштовувати алгоритм сегментації залежно від особливостей зображення. Вибір початкової точки та налаштування порогу впливають на розмір і кількість сегментів, надаючи можливість досягати різних результатів сегментації — від дрібних, детальних сегментів до великих, менш точних.

Завдання 5

Реалізувати сегментацію зображень за допомогою алгоритму зсуву середнього (наприклад, використовуючи квадратні/кубічні вікна в просторі ознак, що попарно не перетинаються). Для цього побудувати простір ознак (багатовимірна гістограма), та виконати зсув середнього (наприклад, попередньо розбивши простір ознак на вікна, та виконуючи зсув середнього для кожного вікна, об’єднуючи їх у випадку співпадіння середніх), після чого об’єднати ознаки, відповідні фінальні середні для яких є близькими у класи еквівалентності. Після цього для отримання безпосередньо сегментації може

бути застосований алгоритм вирощування насіння (із отриманими класами еквівалентності). В якості ознак можна скористатися локальним середнім та стандартним відхиленням (у випадку півтонових зображень), або і безпосередньо значеннями каналів (для кольорового зображення).

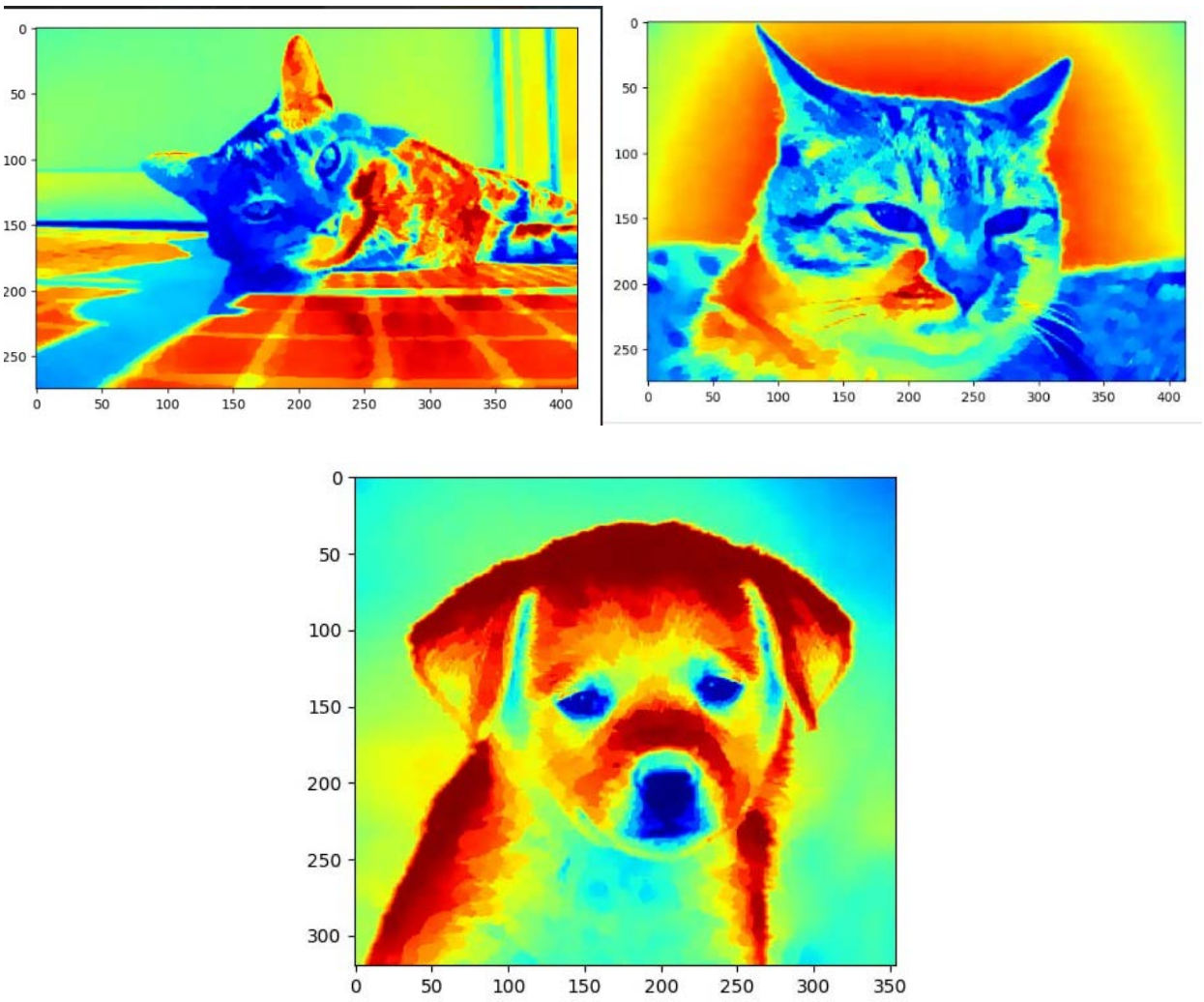
У даному завданні була реалізована процедура сегментації зображень, використовуючи алгоритм зсуву середнього (mean shift). Алгоритм зсуву середнього (mean shift) використовується для сегментації зображень, об’єднуючи пікселі з подібними просторовими та кольоровими характеристиками. Для кожного пікселя обчислюється середнє значення в околі, визначеному параметрами spatial_radius (просторова близькість) і color_radius (різниця інтенсивностей). Процес повторюється до збіжності середніх значень або досягнення максимальної кількості ітерацій.

Результатом є сегментоване зображення, де схожі пікселі групуються в класи. Алгоритм чутливий до параметрів: менші радіуси дають деталізовану сегментацію, а більші — згладжену. Часова складність є недоліком, особливо для великих зображень.

Візуалізація показала, що алгоритм ефективно виділяє області, але точність залежить від параметрів. Для покращення можна застосувати алгоритм вирощування насіння, щоб об’єднати класи в більші сегменти. Алгоритм добре підходить для півтонових і кольорових зображень, але вимагає оптимального налаштування для різних задач.

Результати виконання:

spatial_radius=5, color_radius=10, max_iter=100, epsilon=1



Завдання 6

Реалізувати сегментацію зображень за допомогою алгоритму поширення довіри (член даних, параметр члена гладкості Поттса та критерій зупинки обрати самостійно).

Завдання полягає в реалізації сегментації зображень за допомогою алгоритму поширення довіри (confidence propagation), який використовує параметри члена даних, гладкості Поттса і критеріїв зупинки. Алгоритм базується на оптимізації потенційної енергії між даними та гладкістю, щоб правильно класифікувати пікселі в кілька класів.

Процес починається з ініціалізації матриці довіри, яка представляє ймовірність кожного пікселя належати до кожного класу. Ці ймовірності ініціалізуються випадковим чином, а потім нормалізуються для кожного пікселя, щоб сума

ймовірностей по всіх класах дорівнювала 1. Параметр num_classes визначає кількість класів, на які зображення буде розділено, в той час як image.shape визначає розміри зображення (висоту та ширину).

Далі розраховується термін даних (data term), який оцінює, наскільки кожен піксель у зображенні схожий на певні значення інтенсивності, відповідні класам. Це здійснюється через квадрат різниці між інтенсивністю пікселя та певними мітками класів, що рівномірно розподілені по діапазону інтенсивностей від 0 до 255. Оскільки метою є мінімізація цієї різниці, термін даних від'ємний, що дозволяє алгоритму віддавати перевагу класам, які найближчі до значення пікселя.

Щоб забезпечити гладкість сегментації, використовується термін гладкості (smoothness term), який базується на згортці матриці довіри з фільтром. Це дозволяє пікселям, які є сусідами, мати схожі ймовірності належати до одного класу, тим самим покращуючи цілісність сегментації. Для цього використовуються параметри фільтрації та коефіцієнт гладкості beta, який контролює силу регуляризації.

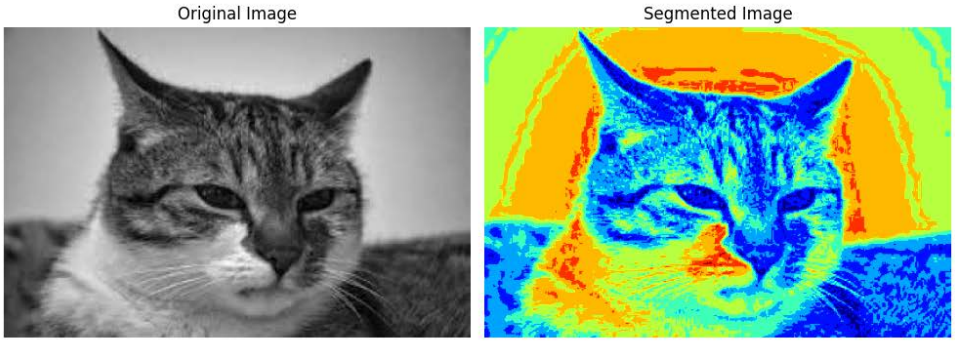
Алгоритм поширення довіри працює таким чином, що для кожної ітерації оновлюється матриця довіри, де кожен піксель отримує нові ймовірності для кожного класу на основі терміну даних та терміну гладкості. Ймовірності нормалізуються після кожного оновлення, щоб сума ймовірностей для кожного пікселя залишалася рівною 1. Процес триває до досягнення критеріїв зупинки: або після досягнення максимального числа ітерацій, або якщо зміни між оновленими та попередніми ймовірностями стають меншими за поріг tol, що вказує на сходження алгоритму.

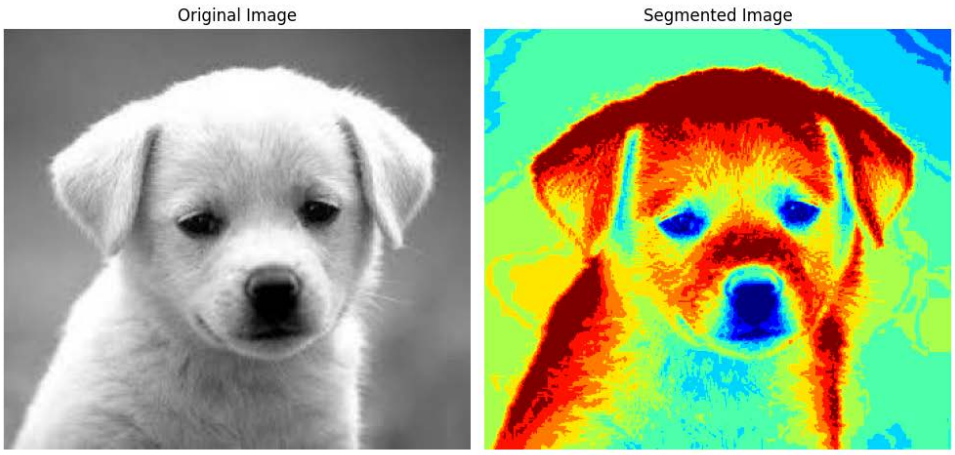
Результатом виконання алгоритму є сегментоване зображення, де кожен піксель віднесений до одного з класів, визначених на початку. Класи визначаються на основі ймовірностей, і кінцеве рішення приймається на основі максимальних ймовірностей для кожного пікселя.

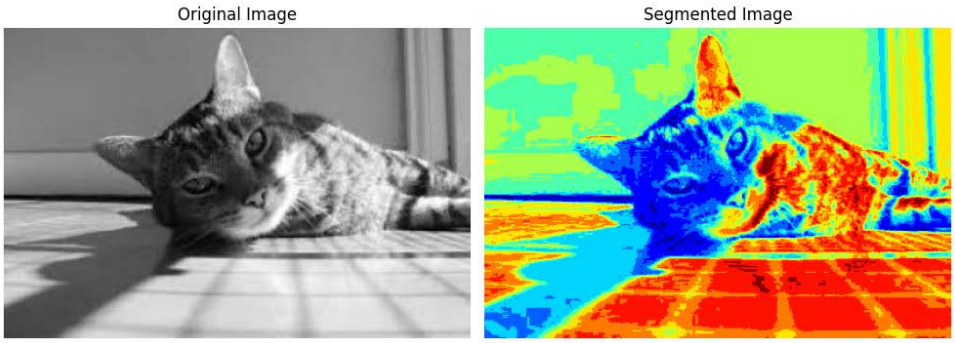
Цей процес можна візуалізувати для кількох зображень, щоб показати як змінюється сегментація в залежності від параметрів алгоритму. Кожне зображення відображається разом з сегментованим варіантом, що дає змогу порівняти результат до та після обробки. Візуалізація допомагає оцінити ефективність сегментації та можливі проблеми, наприклад, через неправильний вибір кількості класів або параметрів гладкості.

Алгоритм ефективний для задач, де потрібно чітко відокремити різні області на зображенні, зокрема в умовах шуму або неоднотипної інтенсивності.

Результати виконання:



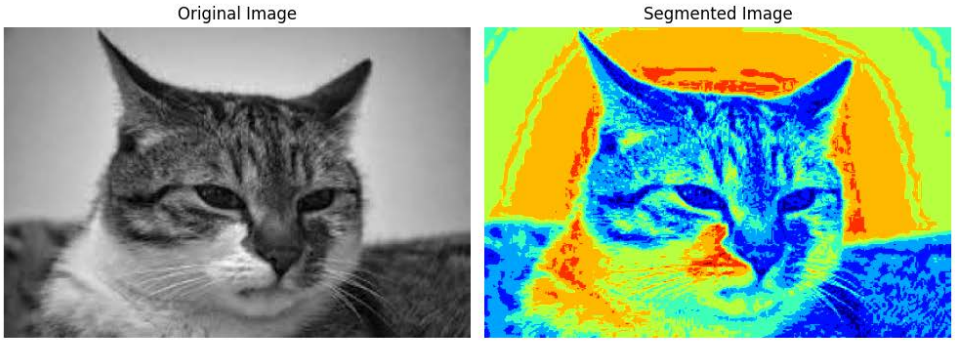


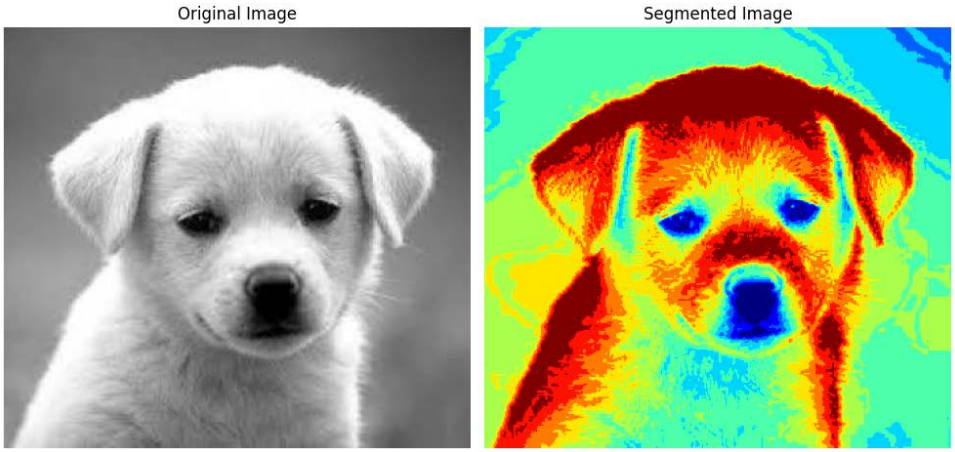


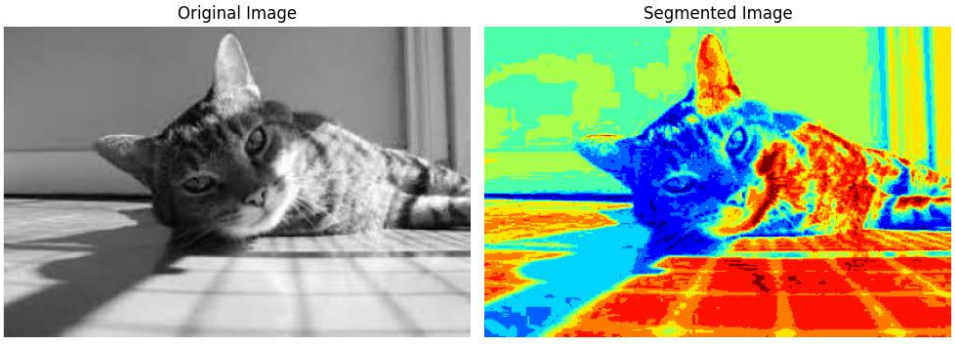
Додаткове завдання:

Дослідити вплив членів даних, членів гладкості, критеріїв зупинки при сегментації зображень за допомогою алгоритму поширення довіри.

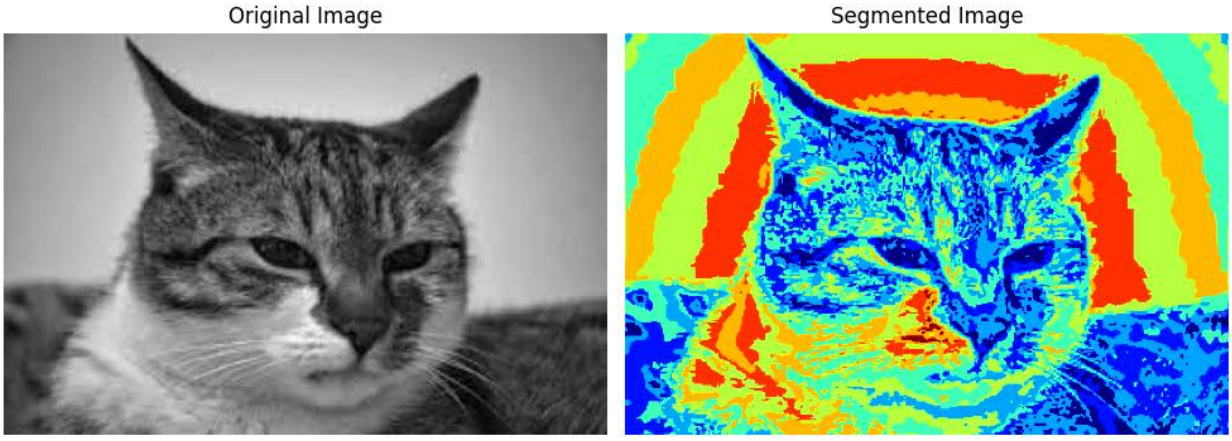
num_classes=3, beta=20, max_iters=50, tol=1e-4

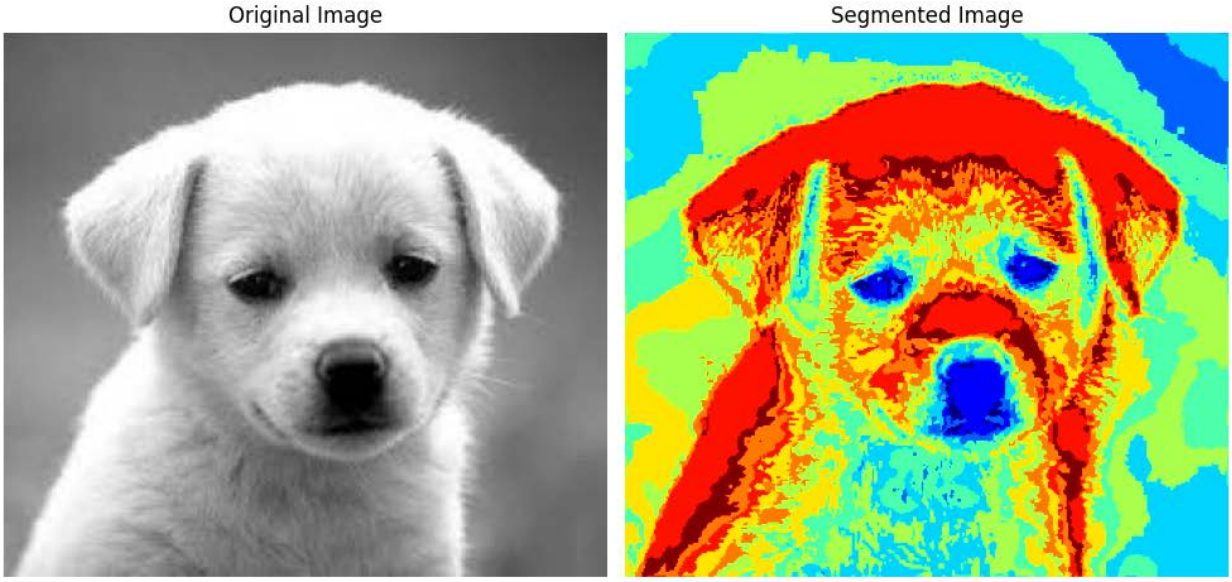


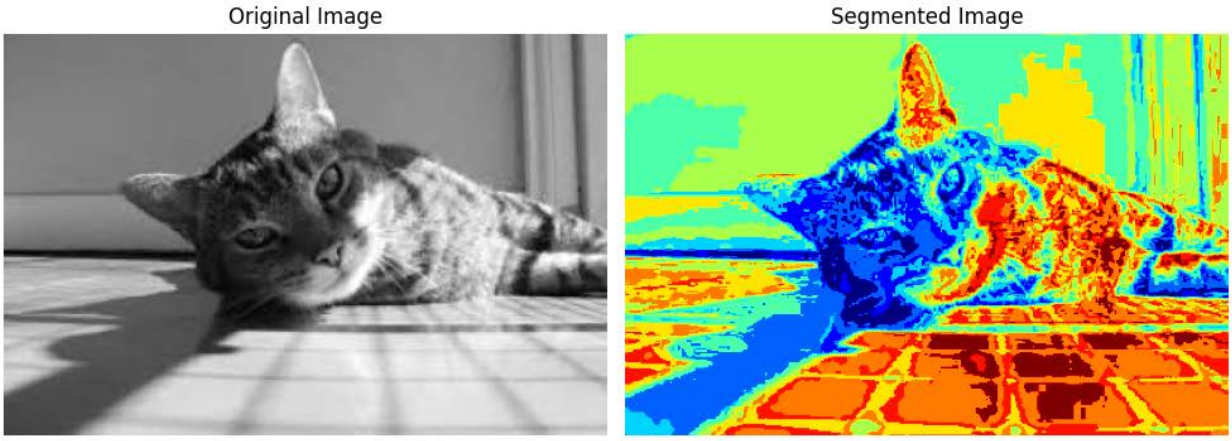




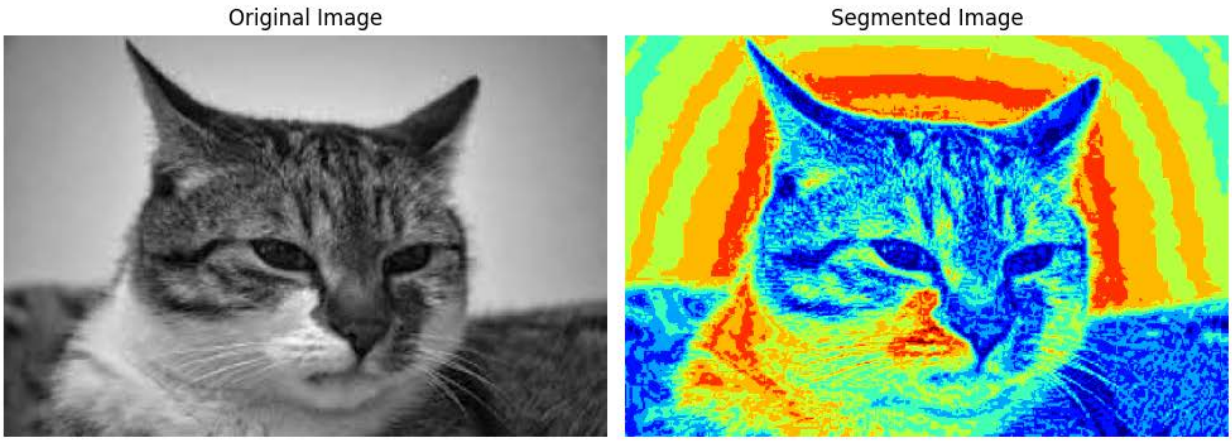
num_classes=5, beta=100, max_iters=100, tol=1e-4

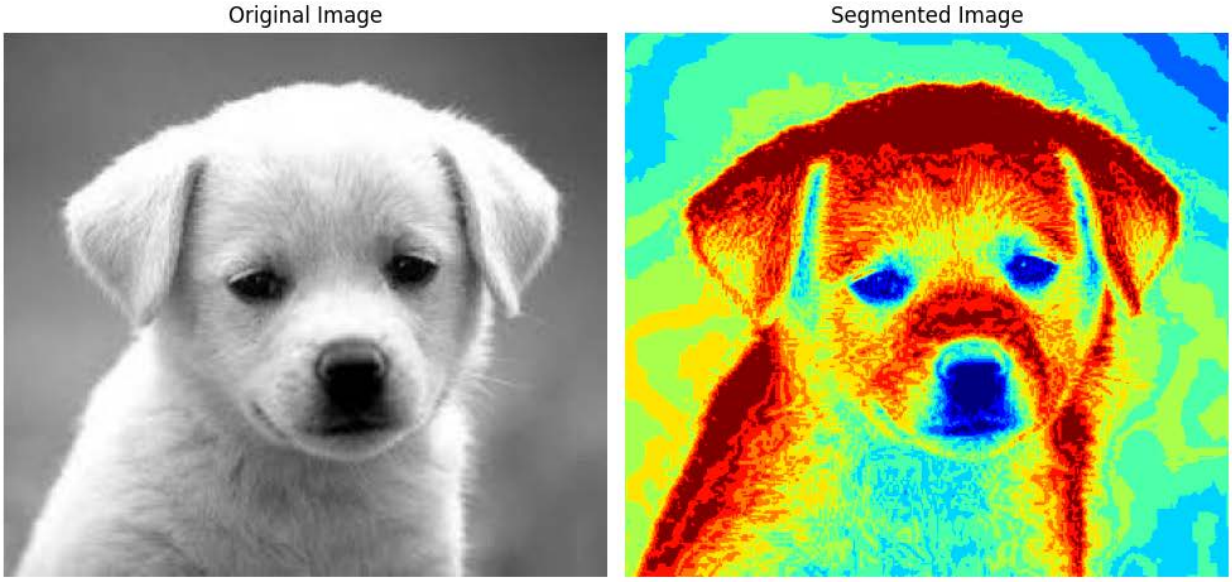


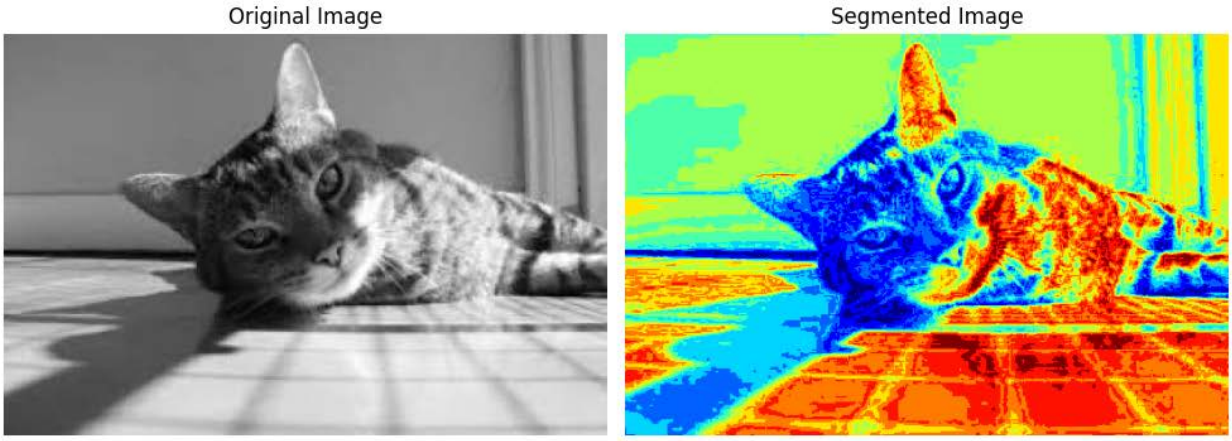




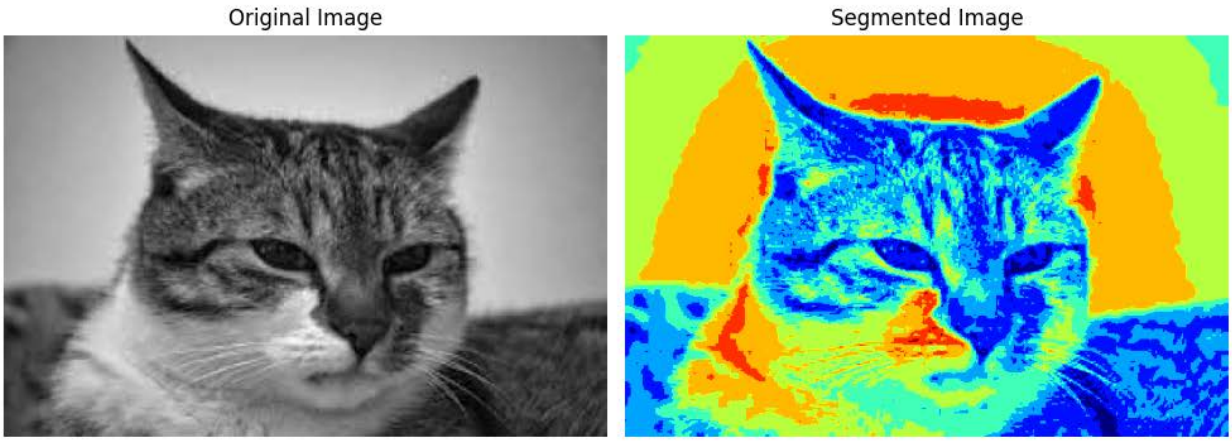
num_classes=5, beta=50, max_iters=2, tol=1e-4

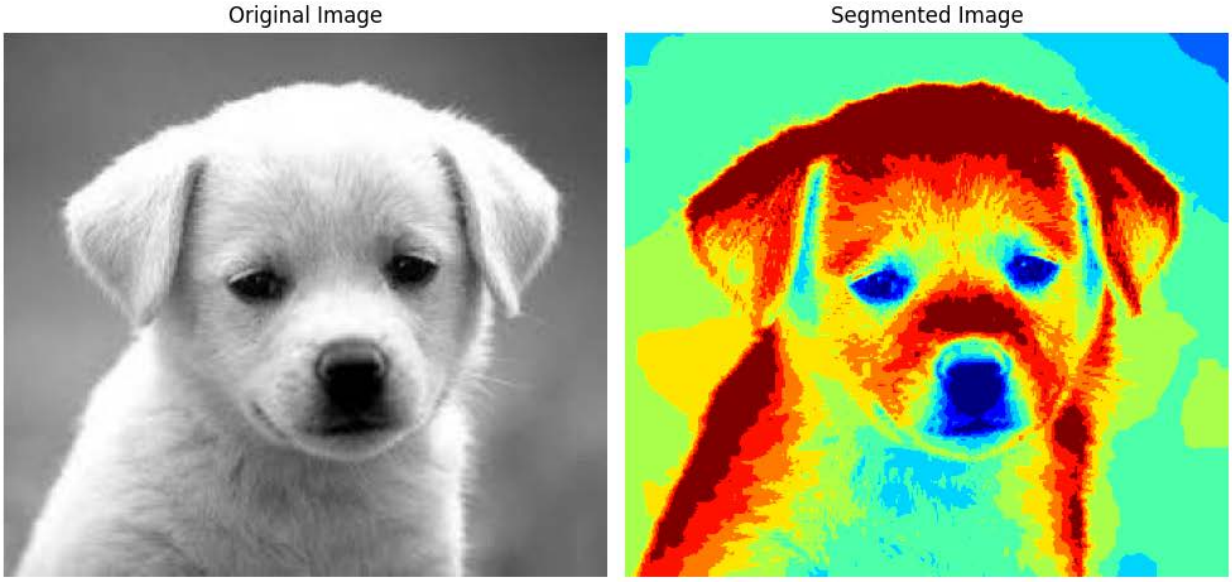


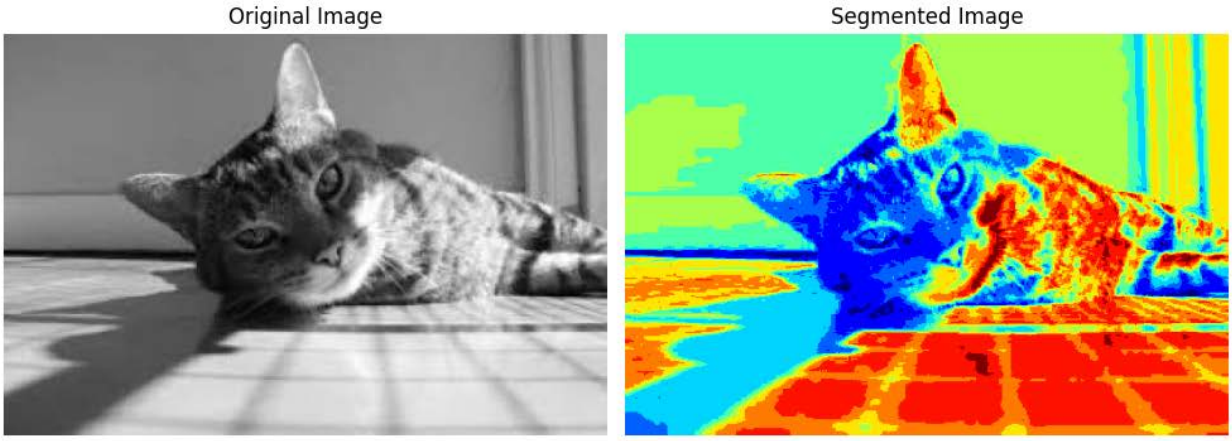




num_classes=7, beta=30, max_iters=75, tol=1e-2







Зміна кожного з параметрів алгоритму поширення довіри безпосередньо впливає на результат сегментації зображення. Кількість класів (num_classes)

визначає, на скільки областей буде розподілено зображення. Якщо число класів велике, то сегментація може стати дуже деталізованою, але це може призвести до більшої складності в обробці та меншої стійкості до шуму. З іншого боку, зменшення кількості класів може спростити сегментацію, але може упустити важливі деталі, якщо на зображенні є багато різних об'єктів або текстур.

Зміна параметра гладкості (beta) контролює, наскільки сильно алгоритм буде прагнути до гладкості між сусідніми пікселями. Якщо значення beta велике, сегментація буде більш гладкою, тобто пікселі, що знаходяться поруч, будуть більш схожі один на одного за класом. Це може бути корисно для зображень з чіткими межами між областями. Однак, якщо beta занадто велике, то можна втратити важливі деталі зображення, особливо на межах об'єктів. Якщо beta мале, тоді сегментація буде менш гладкою, і можуть з'явитися більш різкі межі між областями.

Максимальна кількість ітерацій (max_iters) вказує на кількість циклів, протягом яких алгоритм оновлює ймовірності для кожного пікселя. Чим більше ітерацій, тим точніше може бути сегментація, але і час виконання буде більшим. Якщо ітерацій недостатньо, то алгоритм може не встигнути досягти оптимального результату. Велика кількість ітерацій дозволяє алгоритму точніше адаптувати ймовірності до даних, але зростає ризик перенавчання або тривалого часу виконання.

Поріг зупинки (tol) визначає, на яких умовах алгоритм припинить роботу. Якщо зміни між двома послідовними ітераціями менші за цей поріг, алгоритм зупиниться. Мале значення tol змушує алгоритм завершитися лише після того, як зміни між ітераціями стануть дуже малими, що може привести до точнішої сегментації, але знову ж таки за рахунок більшого часу обчислень. Вищі значення порогу зупинки дозволяють алгоритму завершитися швидше, однак при цьому можливі менш точні результати сегментації, оскільки алгоритм може зупинитися до того, як досягне оптимуму.

Зміна кожного з цих параметрів допомагає знаходити баланс між швидкістю виконання алгоритму та точністю сегментації, а також дозволяє адаптувати процес до різних типів зображень. Тому варто проводити експерименти з різними значеннями цих параметрів, щоб знайти найкращу комбінацію для конкретної задачі.

Висновок:

У цій роботі було реалізовано кілька важливих методів сегментації зображень, кожен з яких має свої особливості та призначення для різних типів зображень та завдань. Спочатку була реалізована процедура бінаризації зображень за заданим глобальним порогом, що дозволяє розділяти півтонові зображення на два класи – об'єкти та фон. Для кожного зображення було підібрано оптимальний поріг, щоб виділити найбажаніші об'єкти, що дозволило гнучко працювати з різними типами зображень.

Далі було використано метод Оцу для автоматичного визначення порогу бінаризації. Алгоритм Оцу шукає оптимальний поріг, що мінімізує варіацію між двома класами (об'єкти і фон) у півтоновому зображенні, що дозволяє підвищити якість сегментації без необхідності вручну вибирати поріг. Порівняння результатів з підібраним порогом показало, що метод Оцу є ефективним для автоматичної бінаризації, хоча для специфічних випадків може бути доцільно застосовувати ручний підхід.

Наступним етапом стало використання стилізації Віннемьоллера для бінаризації зображень, що дозволяє здійснити сегментацію, враховуючи текстурні характеристики. Цей метод дає змогу отримати контури об'єктів з особливою увагою до деталі текстур і контрастів на зображенні. Стилізація Віннемьоллера підходить для випадків, коли необхідно зберегти важливі текстурні елементи при сегментації.

Далі була реалізована сегментація зображень методом вирощування насіння. Цей підхід дозволяє сегментувати зображення на основі певних

ініціалізованих точок (насіння), які поширюються на сусідні пікселі на основі певного відношення еквівалентності, наприклад, інтервалів яскравості. Метод виявився ефективним для простих випадків сегментації, але для складних зображень, де об'єкти мають схожі інтенсивності, потрібно доповнювати алгоритм іншими методами для досягнення точніших результатів.

Наступним етапом була реалізація сегментації методом зсуву середнього, який включає побудову простору ознак та застосування зсуву середнього для кожного вікна. Цей підхід дозволяє більш точно визначити межі між різними об'єктами на зображенні, працюючи з локальними середніми та стандартними відхиленнями пікселів, що дозволяє враховувати варіації в інтенсивності та контрастах. Алгоритм забезпечує точну класифікацію пікселів за класами, що покращує сегментацію для зображень з варіативними текстурами.

Останнім методом, реалізованим у роботі, став алгоритм поширення довіри. Цей метод використовує ймовірності для кожного пікселя, поступово поширюючи їх на сусідні пікселі за допомогою гладкості, що дозволяє досягти високої якості сегментації. Вибір параметрів для члена даних та гладкості, а також критеріїв зупинки відіграють важливу роль в ефективності цього методу. Застосування різних значень цих параметрів впливає на результат, тому важливо ретельно налаштовувати ці значення для конкретних завдань.

Виконане порівняння реалізованих алгоритмів з методами, доступними в OpenCV, дозволило оцінити ефективність запропонованих методів у порівнянні з класичними підходами, такими як поріг Оцу. Це дало змогу підтвердити, що для складних зображень, де присутні не тільки чіткі межі між об'єктами, а й варіативність текстур, запропоновані методи забезпечують кращу сегментацію.

Вплив позицій випадкових початкових точок у методі вирощування насіння та параметрів у методі поширення довіри показав, що на результат сегментації можуть значно впливати початкові умови, тому важливо враховувати ці аспекти при розробці алгоритмів сегментації.

Таким чином, проведене дослідження показало, що різні методи сегментації мають свої переваги та недоліки, і вибір оптимального методу залежить від конкретних характеристик зображень і вимог до результату.