**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**
**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**
**імені ІГОРЯ СІКОРСЬКОГО»**

**Факультет прикладної математики**

**Кафедра системного програмування і спеціалізованих комп'ютерних систем**

«На правах рукопису»
УДК 004.05

До захисту допущено:
Завідувач кафедри
____ Віталій РОМАНКЕВИЧ
«___»_____2021 р.

**Магістерська дисертація**

**на здобуття ступеня магістра**

**за освітньо-професійною програмою**

**«Системне програмування і спеціалізовані комп'ютерні системи»**

**зі спеціальності 123 «Комп'ютерна інженерія»**

# на тему: «Метод розпізнавання меж об'єктів за допомогою модифікованого алгоритму Канні»

Виконала:
студентка ІІ курсу, групи КВ-02мп
Празднікова Маргарита Олександрівна _____

Науковий керівник:
Доцент кафедри СПСКС, к.т.н., доцент,
Потапова Катерина Романівна _____

Рецензент:
Доцент кафедри ПМА, к.т.н., доцент,
Вовк Лілія Борисівна

_____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____

Київ – 2021 року

**Національний технічний університет України**

**«Київський політехнічний інститут імені Ігоря Сікорського»**

**Факультет прикладної математики**

**Кафедра системного програмування і спеціалізованих комп‘ютерних систем**

Рівень вищої освіти – другий (магістерський)

Спеціальність – 123 «Комп‘ютерна інженерія»

Освітньо-професійна програма «Системне програмування і спеціалізовані комп‘ютерні системи»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Віталій РОМАНКЕВИЧ

«___»_____2020 р.

**ЗАВДАННЯ**
**на магістерську дисертацію студенту**

**Праздніковій Маргариті Олександрівні**

1. Тема дисертації «Метод розпізнавання меж об'єктів за допомогою модифікованого алгоритму Канні», науковий керівник дисертації Потапова Катерина Романівна, к.т.н., доцент, затверджені наказом по університету від «5» 11. 2021 р. №3682-С

2. Термін подання студентом дисертації 7.12.2021

3. Об’єкт дослідження: алгоритми розпізнання меж об’єктів.

4. Вихідні дані: модифікований алгоритм та порівняння його роботи з існуючимии.

5. Перелік завдань, які потрібно розробити:
   - опис предметної області досліджень та аналіз існуючих інструментів для статичного аналізу коду;
   - модифікація алгоритму Канні;
   - програмна реалізація та порівняння з існуючими алгоритмами.

6. Перелік графічного матеріалу – презентація.

7. Перелік публікацій:

- XIV науковій конференції магістрантів та аспірантів «Прикладна математика та комп'ютинг» ПМК-2021 (Київ, 17-19 листопада 2021 р.)

- V Міжнародна науково-практична конференція "MODERN SCIENTIFIC RESEARCH: ACHIEVEMENTS, INNOVATIONS AND DEVELOPMENT PROSPECTS", 24-26 жовтня 2021 Берлін, Німеччина..

9. Дата видачі завдання 7.10.2020

Календарний план

| № з/п | Назва етапів виконання магістерської дисертації | Термін виконання етапів магістерської дисертації | Примітка |
|---|---|---|---|
| 1 | Вивчення літератури за тематикою проекту | 15.10.2020 | |
| 2 | Аналіз існуючих методів та алгоритмів | 29.10.2020 | |
| 3 | Визначення структури магістерської дисертації та змісту пояснювальної записки. | 10.11.2020 | |
| 4 | Робота над першим розділом магістерської дисертації; | 05.02.2021 | |
| 5 | Аналіз способів та алгоритмів для обробки зображення, робота над другим розділом. | 13.03.2021 | |
| 6 | Модифікація алгоритму Канні, робота над третім розділом. | 08.04.2021 | |
| 7 | Програмна реалізація виводу роботи модифіковаанного алгоритму Канні. | 21.09.2021 | |
| 8 | Оформлення текстової і графічної частини магістерської дисертації | 10.10.2021 | |
| 9 | Попередній розгляд магістерської дисертації на кафедрі | 01.12.2021 | |

Студентка Маргарита ПРАЗДНІКОВА

Науковий керівник Катерина ПОТАПОВА

# РЕФЕРАТ

**Актуальність теми.** Пошук меж об’єктів в потоку чи на окремому зображенні використовується в багатьох галузях діяльності та буденних справах людини. Наприклад, в медицині за допомогою такої обробки можна визначити пухлину чи тріщину на рентгенівському знімку, визначити межі та сфокусувати прилад, що веде зйомку, на потрібній ділянці. Камери відеоспостереження за транспортом за допомогою таких технологій можуть визначити ділянку, на якій знаходиться номер машини, що порушила правила, для подальшої ідентифікації. Так як машина рухається, то в моменті стоп-кадру номер автівки і обличчя людини нечіткі. Отже, необхідно спершу визначити ту частину зображення, яку слід обробити щільніше, пом’якшити тон та покращити якість картинки. Також, покращення визначення границь зображення може використовуватися для ідентифікації конкретних написів на стінах або плакатах, що необхідно для комерційних цілей маркетингових фірм.

**Об’єктом дослідження** є метод розпізнавання меж об'єктів за алгоритму Канні.

**Предметом дослідження** є метод розпізнавання меж об'єктів за допомогою алгоритму Канні.

**Мета роботи:** модифікація алгоритму Канні для знаходження меж об’єкту.

**Наукова новизна** полягає в наступному: запропоновано модифікацію алгоритму Канні. Особливості:

- Використано розглядання сусідніх пікселі на зображенні;
- Використано предикат для визначення неявних меж об’єктів;
- Використано багатопоточність для пришвидшення виконання операцій обчислення.

**Практична цінність** пришвидшене визначення меж об’єктів на зображенні.

**Апробація роботи.** Основні положення і результати роботи були представлені та обговорювались на 2 конференціях, а саме: XIV науковій конференції магістрантів та аспірантів «Прикладна математика та комп'ютинг» ПМК-2021 (Київ, 17-19 листопада 2021 р.) та V Міжнародна науково-практична конференція "MODERN SCIENTIFIC RESEARCH: ACHIEVEMENTS, INNOVATIONS AND DEVELOPMENT PROSPECTS", 24-26 жовтня 2021 Берлін, Німеччина..

**Структура та обсяг роботи.** Магістерська дисертація складається з вступу, чотрьох розділів та висновків.

*У вступі* подано загальну характеристику. Обґрунтовано актуальність напрямку досліджень, сформульовано мету і задачі досліджень, показано наукову новизну отриманих результатів і практичну цінність роботи, наведено відомості про апробацію результатів і їхнє впровадження.

*У першому розділі* йде мова про теоретичні відомості, котрі необхідні для роботи та розглядається необхідність в данній сфері.

*У другому розділі* проаналізовано існуючі методи та алгоритми для обробки зображень. А також проведений аналіз, який дає змогу визначити основні їх переваги та недоліки. Наведено практичне застосування цих методів та алгоритмів.

*У третьому розділі* описується модифікації алгоритму Канні для покращення та прискорення його роботи.

*У третьому розділі* проводиться якісний аналіз та порівняння модифікованого алгоритму Канні та існуючими.

*У висновках* представлені результати проведеної роботи.

Робота представлена на 93 аркушах, містить посилання на список використаних літературних джерел.

**Ключові слова**: сегментація, межі зображення, демонстраційна плата, алгоритм Канні, градієнт, градація сірого кольору.

# РЕФЕРАТ

**Актуальность темы.** Поиск границ объектов в потоке или на отдельном изображении используется во многих сферах деятельности и будничных делах человека. Например, в медицине с помощью такой обработки можно определить опухоль или трещину на рентгеновском снимке, определить границы и сфокусировать ведущий съемку прибор на нужном участке. Камеры видеонаблюдения за транспортом с помощью таких технологий могут определить участок, на котором находится номер нарушившей правила машины для дальнейшей идентификации. Так как машина двигается, то в моменте стоп-кадра номер автомобиля и лица человека нечетки. Следовательно, необходимо сначала определить ту часть изображения, которую следует обработать более плотно, смягчить тон и улучшить качество картинки.

**Объектом исследования** является метод распознавания между объектами по алгоритму Канни.

**Предметом исследования** является метод распознавания между объектами с помощью алгоритма Канни.

**Цель работы:** модификация метода Канни для нахождения меж объектом.

**Научная новизна состоит в следующем**: предложена модификация алгоритма Канни. Особенности:

- Использовано рассмотрение соседних пикселей на изображении;
- Использован предикат для определения неявных границ объектов;
- Использовано многопоточность для ускорения выполнения операций вычисления.

**Практическая ценность** ускоренное определения между объектами на изображении.

**Апробация работы.** Основные положения и результаты работы были представлены и обсуждались на 2 конференциях, а именно: XIV научной конференции магистрантов и аспирантов «Прикладная математика и

компьютер» ПМК-2021 (Киев, 17-19 ноября 2021 г.) и V Международная научно-практическая конференция “MODERN SCIENTIFIC RESEARCH: ACHIEVEMENTS, INNOVATIONS AND DEVELOPMENT PROSPECTS”, 24-26 октября 2021 г. Берлин, Германия.

**Структура и размер работы.** Магистерская диссертация состоит из введения, четырех глав и выводов.

*Во вступлении* дана общая характеристика. Обоснована актуальность направления исследований, сформулированы цели и задачи исследований, показана научная новизна полученных результатов и практическая ценность работы, приведены сведения об апробации результатов и их внедрении.

*В первой главе* говорится о теоретических сведениях, необходимых для работы и рассматривается необходимость в данной сфере.

*В другом разделе* проанализированы существующие методы и алгоритмы обработки изображений. А также проведен анализ, позволяющий определить основные их преимущества и недостатки. Представлено практическое применение этих методов и алгоритмов.

*В третьей* главе описывается модификация алгоритма Канни для улучшения и ускорения его работы.

*В четвертой главе* производится качественный анализ и сравнение модифицированного алгоритма Канни и существующими.

*В выводах* представлены результаты проделанной работы.

Работа представлена на 85 листах, содержащая ссылку на список использованных литературных источников.

**Ключевые слова**: сегментация, границы изображения, демонстрационная плата, алгоритм Канни, градиент, градация серого цвета.

## ABSTRACT

**Actuality of theme.** Finding the boundaries of objects in a stream or in a single image is used in many areas of activity and everyday human affairs. For example, in medicine, this treatment can identify a tumor or crack on an X-ray, define boundaries, and focus the imaging device on the desired area. Surveillance cameras with the help of such technologies can determine the area where the number of the car that violated the rules is located, for further identification. Therefore, you must first determine the part of the image that should be processed more densely, soften the tone and improve the picture quality.

**The object of study** is a study of algorithms for finding the boundaries of objects on a photograph and a modification of the Canney algorithm.

**The subject of research** there is an acceleration and improvement of the Canney algorithm.

**Purpose**: to modify the Canney algorithm to find the boundaries of an object.

**The scientific novelty** of finding the boundaries of objects by modifying the Canney algorithm. Features:

- Used consideration of neighboring pixels in the image;
- Predicate used to define implicit object boundaries;
- Multithreading is used to speed up computational operations.

**The practical value** of quickly defining the boundaries of objects in the image.

**Approbation of work.** The main provisions and results of the work were presented and discussed at 2 conferences, namely: XIV scientific conference of undergraduates and graduate students "Applied Mathematics and Computing" PMK-2021 (Kyiv, November 17-19, 2021) and V International Scientific and Practical Conference “MODERN SCIENTIFIC RESEARCH: ACHIEVEMENTS, INNOVATIONS AND DEVELOPMENT PROSPECTS”, October 24-26, 2021 Berlin, Germany ..

**Structure and scope of work.** The master's dissertation consists of an introduction, four sections and conclusions.

*In the introduction* a general description is given. The relevance of the research direction is substantiated, the purpose and tasks of the research are formulated, the scientific novelty of the obtained results and the practical value of the work are shown, the information on the approbation of the results and their implementation is given.

*The first* chapter talks about the theoretical information required for the job and discusses the need for this area.

*In second section*, the existing methods and algorithms for image processing are analyzed. And also an analysis was carried out to determine their main advantages and disadvantages. The practical application of these methods and algorithms is presented.

*The third chapter* describes a modification of Canny's algorithm to improve and speed up its work.

*In the fourth section* a qualitative analysis and comparison of the modified Canney algorithm with existing ones is performed.

*In the conclusions* the results of the work are presented.

The work is presented on 85 sheets, contains references to the list of used literature sources.

**Keywords**: segmentation, image boundaries, demonstration board, Canney algorithm, gradient, grayscale.

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**
**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**
**імені ІГОРЯ СІКОРСЬКОГО»**

**Факультет прикладної математики**

**Кафедра системного програмування і спеціалізованих комп'ютерних систем**

«На правах рукопису»
УДК 004.05

До захисту допущено:
Завідувач кафедри
____ Віталій РОМАНКЕВИЧ
«___»_____2021 р.

**Магістерська дисертація**

**на здобуття ступеня магістра**

**за освітньо-професійною програмою**

**«Системне програмування і спеціалізовані комп'ютерні системи»**

**зі спеціальності 123 «Комп'ютерна інженерія»**

# на тему: «Метод розпізнавання меж об'єктів за допомогою модифікованого алгоритму Канні»

Виконала:
студентка ІІ курсу, групи КВ-02мп
Празднікова Маргарита Олександрівна _____

Науковий керівник:
Доцент кафедри СПСКС, к.т.н., доцент,
Потапова Катерина Романівна _____

Рецензент:
Доцент кафедри ПМА, к.т.н., доцент,
Вовк Лілія Борисівна

_____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____

Київ – 2021 року

**Національний технічний університет України**

**«Київський політехнічний інститут імені Ігоря Сікорського»**

**Факультет прикладної математики**

**Кафедра системного програмування і спеціалізованих комп‘ютерних систем**

Рівень вищої освіти – другий (магістерський)

Спеціальність – 123 «Комп‘ютерна інженерія»

Освітньо-професійна програма «Системне програмування і спеціалізовані комп‘ютерні системи»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Віталій РОМАНКЕВИЧ

«___»_____2020 р.

**ЗАВДАННЯ**
**на магістерську дисертацію студенту**

**Праздніковій Маргариті Олександрівні**

1. Тема дисертації «Метод розпізнавання меж об'єктів за допомогою модифікованого алгоритму Канні», науковий керівник дисертації Потапова Катерина Романівна, к.т.н., доцент, затверджені наказом по університету від «5» 11. 2021 р. №3682-С

2. Термін подання студентом дисертації 7.12.2021

3. Об’єкт дослідження: алгоритми розпізнання меж об’єктів.

4. Вихідні дані: модифікований алгоритм та порівняння його роботи з існуючимии.

5. Перелік завдань, які потрібно розробити:
   - опис предметної області досліджень та аналіз існуючих інструментів для статичного аналізу коду;
   - модифікація алгоритму Канні;
   - програмна реалізація та порівняння з існуючими алгоритмами.

6. Перелік графічного матеріалу – презентація.

7. Перелік публікацій:

- XIV науковій конференції магістрантів та аспірантів «Прикладна математика та комп'ютинг» ПМК-2021 (Київ, 17-19 листопада 2021 р.)

- V Міжнародна науково-практична конференція "MODERN SCIENTIFIC RESEARCH: ACHIEVEMENTS, INNOVATIONS AND DEVELOPMENT PROSPECTS", 24-26 жовтня 2021 Берлін, Німеччина..

9. Дата видачі завдання 7.10.2020

Календарний план

| № з/п | Назва етапів виконання магістерської дисертації | Термін виконання етапів магістерської дисертації | Примітка |
|---|---|---|---|
| 1 | Вивчення літератури за тематикою проекту | 15.10.2020 | |
| 2 | Аналіз існуючих методів та алгоритмів | 29.10.2020 | |
| 3 | Визначення структури магістерської дисертації та змісту пояснювальної записки. | 10.11.2020 | |
| 4 | Робота над першим розділом магістерської дисертації; | 05.02.2021 | |
| 5 | Аналіз способів та алгоритмів для обробки зображення, робота над другим розділом. | 13.03.2021 | |
| 6 | Модифікація алгоритму Канні, робота над третім розділом. | 08.04.2021 | |
| 7 | Програмна реалізація виводу роботи модифіковаанного алгоритму Канні. | 21.09.2021 | |
| 8 | Оформлення текстової і графічної частини магістерської дисертації | 10.10.2021 | |
| 9 | Попередній розгляд магістерської дисертації на кафедрі | 01.12.2021 | |

Студентка Маргарита ПРАЗДНІКОВА

Науковий керівник Катерина ПОТАПОВА

# РЕФЕРАТ

**Актуальність теми.** Пошук меж об’єктів в потоку чи на окремому зображенні використовується в багатьох галузях діяльності та буденних справах людини. Наприклад, в медицині за допомогою такої обробки можна визначити пухлину чи тріщину на рентгенівському знімку, визначити межі та сфокусувати прилад, що веде зйомку, на потрібній ділянці. Камери відеоспостереження за транспортом за допомогою таких технологій можуть визначити ділянку, на якій знаходиться номер машини, що порушила правила, для подальшої ідентифікації. Так як машина рухається, то в моменті стоп-кадру номер автівки і обличчя людини нечіткі. Отже, необхідно спершу визначити ту частину зображення, яку слід обробити щільніше, пом’якшити тон та покращити якість картинки. Також, покращення визначення границь зображення може використовуватися для ідентифікації конкретних написів на стінах або плакатах, що необхідно для комерційних цілей маркетингових фірм.

**Об’єктом дослідження** є метод розпізнавання меж об'єктів за алгоритму Канні.

**Предметом дослідження** є метод розпізнавання меж об'єктів за допомогою алгоритму Канні.

**Мета роботи:** модифікація алгоритму Канні для знаходження меж об’єкту.

**Наукова новизна** полягає в наступному: запропоновано модифікацію алгоритму Канні. Особливості:

- Використано розглядання сусідніх пікселі на зображенні;
- Використано предикат для визначення неявних меж об’єктів;
- Використано багатопоточність для пришвидшення виконання операцій обчислення.

**Практична цінність** пришвидшене визначення меж об’єктів на зображенні.

**Апробація роботи.** Основні положення і результати роботи були представлені та обговорювались на 2 конференціях, а саме: XIV науковій конференції магістрантів та аспірантів «Прикладна математика та комп'ютинг» ПМК-2021 (Київ, 17-19 листопада 2021 р.) та V Міжнародна науково-практична конференція "MODERN SCIENTIFIC RESEARCH: ACHIEVEMENTS, INNOVATIONS AND DEVELOPMENT PROSPECTS", 24-26 жовтня 2021 Берлін, Німеччина..

**Структура та обсяг роботи.** Магістерська дисертація складається з вступу, чотрьох розділів та висновків.

*У вступі* подано загальну характеристику. Обґрунтовано актуальність напрямку досліджень, сформульовано мету і задачі досліджень, показано наукову новизну отриманих результатів і практичну цінність роботи, наведено відомості про апробацію результатів і їхнє впровадження.

*У першому розділі* йде мова про теоретичні відомості, котрі необхідні для роботи та розглядається необхідність в данній сфері.

*У другому розділі* проаналізовано існуючі методи та алгоритми для обробки зображень. А також проведений аналіз, який дає змогу визначити основні їх переваги та недоліки. Наведено практичне застосування цих методів та алгоритмів.

*У третьому розділі* описується модифікації алгоритму Канні для покращення та прискорення його роботи.

*У третьому розділі* проводиться якісний аналіз та порівняння модифікованого алгоритму Канні та існуючими.

*У висновках* представлені результати проведеної роботи.

Робота представлена на 93 аркушах, містить посилання на список використаних літературних джерел.

**Ключові слова**: сегментація, межі зображення, демонстраційна плата, алгоритм Канні, градієнт, градація сірого кольору.

## РЕФЕРАТ

**Актуальность темы.** Поиск границ объектов в потоке или на отдельном изображении используется во многих сферах деятельности и будничных делах человека. Например, в медицине с помощью такой обработки можно определить опухоль или трещину на рентгеновском снимке, определить границы и сфокусировать ведущий съемку прибор на нужном участке. Камеры видеонаблюдения за транспортом с помощью таких технологий могут определить участок, на котором находится номер нарушившей правила машины для дальнейшей идентификации. Так как машина двигается, то в моменте стоп-кадра номер автомобиля и лица человека нечетки. Следовательно, необходимо сначала определить ту часть изображения, которую следует обработать более плотно, смягчить тон и улучшить качество картинки.

**Объектом исследования** является метод распознавания между объектами по алгоритму Канни.

**Предметом исследования** является метод распознавания между объектами с помощью алгоритма Канни.

**Цель работы:** модификация метода Канни для нахождения меж объектом.

**Научная новизна состоит в следующем**: предложена модификация алгоритма Канни. Особенности:

- Использовано рассмотрение соседних пикселей на изображении;
- Использован предикат для определения неявных границ объектов;
- Использовано многопоточность для ускорения выполнения операций вычисления.

**Практическая ценность** ускоренное определения между объектами на изображении.

**Апробация работы.** Основные положения и результаты работы были представлены и обсуждались на 2 конференциях, а именно: XIV научной конференции магистрантов и аспирантов «Прикладная математика и

компьютер» ПМК-2021 (Киев, 17-19 ноября 2021 г.) и V Международная научно-практическая конференция “MODERN SCIENTIFIC RESEARCH: ACHIEVEMENTS, INNOVATIONS AND DEVELOPMENT PROSPECTS”, 24-26 октября 2021 г. Берлин, Германия.

**Структура и размер работы.** Магистерская диссертация состоит из введения, четырех глав и выводов.

*Во вступлении* дана общая характеристика. Обоснована актуальность направления исследований, сформулированы цели и задачи исследований, показана научная новизна полученных результатов и практическая ценность работы, приведены сведения об апробации результатов и их внедрении.

*В первой главе* говорится о теоретических сведениях, необходимых для работы и рассматривается необходимость в данной сфере.

*В другом разделе* проанализированы существующие методы и алгоритмы обработки изображений. А также проведен анализ, позволяющий определить основные их преимущества и недостатки. Представлено практическое применение этих методов и алгоритмов.

*В третьей* главе описывается модификация алгоритма Канни для улучшения и ускорения его работы.

*В четвертой главе* производится качественный анализ и сравнение модифицированного алгоритма Канни и существующими.

*В выводах* представлены результаты проделанной работы.

Работа представлена на 85 листах, содержащая ссылку на список использованных литературных источников.

**Ключевые слова**: сегментация, границы изображения, демонстрационная плата, алгоритм Канни, градиент, градация серого цвета.

## ABSTRACT

**Actuality of theme.** Finding the boundaries of objects in a stream or in a single image is used in many areas of activity and everyday human affairs. For example, in medicine, this treatment can identify a tumor or crack on an X-ray, define boundaries, and focus the imaging device on the desired area. Surveillance cameras with the help of such technologies can determine the area where the number of the car that violated the rules is located, for further identification. Therefore, you must first determine the part of the image that should be processed more densely, soften the tone and improve the picture quality.

**The object of study** is a study of algorithms for finding the boundaries of objects on a photograph and a modification of the Canney algorithm.

**The subject of research** there is an acceleration and improvement of the Canney algorithm.

**Purpose**: to modify the Canney algorithm to find the boundaries of an object.

**The scientific novelty** of finding the boundaries of objects by modifying the Canney algorithm. Features:

- Used consideration of neighboring pixels in the image;
- Predicate used to define implicit object boundaries;
- Multithreading is used to speed up computational operations.

**The practical value** of quickly defining the boundaries of objects in the image.

**Approbation of work.** The main provisions and results of the work were presented and discussed at 2 conferences, namely: XIV scientific conference of undergraduates and graduate students "Applied Mathematics and Computing" PMK-2021 (Kyiv, November 17-19, 2021) and V International Scientific and Practical Conference “MODERN SCIENTIFIC RESEARCH: ACHIEVEMENTS, INNOVATIONS AND DEVELOPMENT PROSPECTS”, October 24-26, 2021 Berlin, Germany ..

**Structure and scope of work.** The master's dissertation consists of an introduction, four sections and conclusions.

*In the introduction* a general description is given. The relevance of the research direction is substantiated, the purpose and tasks of the research are formulated, the scientific novelty of the obtained results and the practical value of the work are shown, the information on the approbation of the results and their implementation is given.

*The first* chapter talks about the theoretical information required for the job and discusses the need for this area.

*In second section*, the existing methods and algorithms for image processing are analyzed. And also an analysis was carried out to determine their main advantages and disadvantages. The practical application of these methods and algorithms is presented.

*The third chapter* describes a modification of Canny's algorithm to improve and speed up its work.

*In the fourth section* a qualitative analysis and comparison of the modified Canney algorithm with existing ones is performed.

*In the conclusions* the results of the work are presented.

The work is presented on 85 sheets, contains references to the list of used literature sources.

**Keywords**: segmentation, image boundaries, demonstration board, Canney algorithm, gradient, grayscale.

Зміст

# СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

RGB – кольорова модель RGB це адитивна колірна модель[1], в якій червоний, зелений і синій основні кольори світла додаються різними способами для відтворення широкого спектру кольорів. Назва моделі походить від ініціалів трьох адитивних основних кольорів: червоного, зеленого та синього.

Сегментація — у загальному значенні — процес або результат розподілу чого-небудь на окремі частини, сегменти.

Розмивання Гауса — це метод фільтрації зображення за допомогою функції Гауса, який призводить до розмивання зображення.

Фільтр Гауса — це фільтр, імпульсна характеристика якого є функцією Гауса (або наближенням до неї, оскільки справжня гауссова характеристика має нескінченну імпульсну характеристику). Фільтри Гаусса мають властивості не мати перевищення на вхід ступінчастої функції, мінімізуючи час наростання та спаду.

YUV — кольорова модель, в якій колір представляється як 3 компоненти — яскравість (Y) і дві кольороворізносні (U і V).

Коефіцієнт — характеристика процесу, явища, речовини або поля, яка має відносно сталий характер.

Піксель (скорочено від англ. PICture'S ELement — елемент зображення) — найдрібніша одиниця цифрового зображення в растровій графіці.

# ВСТУП

У сучасному світі все більше значення займає штучний інтелект та наукові дослідження, пов'язані з його застосуванням, удосконаленням, аналізом алгоритмів та шляхів удосконалення обробки даних.

Комп'ютерний зір – це область штучного інтелекту, яка навчає комп'ютери розуміти та інтерпретувати візуальний світ. Використовуючи цифрові зображення з камер та відео, а також моделі глибокого навчання, машини можуть точно ідентифікувати та класифікувати об'єкти, а потім реагувати на те, що вони бачать.

Однією з основних завдань комп'ютерного зору – завдання ідентифікації об'єкта на зображенні, визначення його меж та характеристик. Це завдання (пошук меж об'єкта та визначення його характеристик) існують у різних сферах діяльності людини. Завдяки очевидній комерціалізації знайдено велику кількість приватних рішень даного завдання, проте побудова універсального рішення не є можливою.

При обробці та аналізі зображень виконуються такі етапи:

- Сегментація (за допомогою алгоритмів виділення контурів та бінаризації);

- пошук, виділення та опис пов'язаних областей;

- порівняння характеристик знайдених об'єктів з еталоном.

Ключовим етапом ідентифікації є порівняння виділеного контуром вихідного зображення з певним еталонним зображенням. Для цього використовуються функції спектральної яскравості, принцип узгодженості оцінок, оцінка максимальної правдоподібності, інші алгоритми.

На етапі опису можуть використовуватись обчислення геометричних ознак, алгоритм випадкової вибірки, аналіз гіперспектральної інформації,

нейронні мережі. Для виділення областей, присутніх на зображенні, використовуються різні алгоритми, такі як розростання регіонів, хвильове вирощування областей, добогання. Дані алгоритми не застосовні в задачах з підвищеною деталізованістю зображень, оскільки основні об'єкти на зображенні розбиваються на десятки дрібніших ділянок, що не піддаються аналізу.

Сегментація зображень починається із застосування окремо або в комбінації методів бінаризації, виділення контурів та придушення «шумів». Від якості виконання цього етапу залежить результат подальшої обробки зображення.

В даний момент для сегментації найчастіше використовуються відомі алгоритми бінаризації: метод Оцу, алгоритми Бредлі-Рота, Ніблека, Саволи та Крістіана Вульфа, а також алгоритм аналізу середнього арифметичного кольору пікселя в моделі RGB.

Відомі і найчастіше застосовуються алгоритми виділення контуру: різні комбінації Лапласіана з Гауссіаном, оператор Робертса, оператор Собеля, оператор Прюїт, оператор Кірша, оператор Канні. Найбільш якісний результат дає використання оператора Лапласіана 5×5 або оператора Канні.

# 1 ТЕОРЕТИЧНІ ВІДОМОСТІ ПРО ОБРОБКУ ГРАФІЧНИХ ФАЙЛІВ

## 1.1 Короткі теоретичні відомості

Комп'ютерна графіка – можливість обробляти фотокартки, зображення, текст та створювати нові зображення, лінії, діаграм тощо. на комп'ютері з допомогою програмування. Усі графічні файли складаються із кількості пікселів. Піксель - це найменше графічне зображення або одиниця, яка представлена на екрані комп'ютера.[28]

Виділяються два типи комп’ютерної графіки: інтерактивна та неінтерактивна комп'ютерні графіки.

Інтерактивна комп'ютерна графіка: передбачає двосторонній зв'язок між комп'ютером та користувачем. Це дає змогу не тільки програмісту, а й користувачу обробляти зображення через інтерфейс програми.

Під час отримання сигналів від пристрою введення, комп'ютер може відповідним чином змінити зображення, що відображається. Користувачеві здається, що зображення миттєво змінюється у відповідь на його команди. Він може дати серію команд, кожна з яких генерує графічну відповідь комп'ютера. Таким чином, він підтримує розмову або діалог з комп'ютером.

Інтерактивна комп'ютерна графіка опосередковано впливає на наше життя. Наприклад, допомагає навчати пілотів наших літаків. Ми можемо створити авіасимулятор, який допоможе пілотам тренуватись не на реальному літаку, а на майданчиках під керуванням авіасимулятора. Симулятор польоту є макет кабіни екіпажу літака, що містить всі звичайні елементи управління і оточений екранами, на яких ми проектуємо види місцевості, що згенеровані комп'ютером, видимої при зльоті і посадці.

Авіаційні тренажери мають багато переваг перед справжніми літаками у навчальних цілях, включаючи економію палива, безпеку та можливість познайомити учня з великою кількістю аеропортів світу.

Неінтерактивна комп'ютерна графіка (також відома як пасивна комп'ютерна графіка) – це комп'ютерна графіка, в якій користувач не має контролю над зображенням. Зображення – це просто продукт статичної збереженої програми і працюватиме відповідно до інструкцій, даних у програмі, лінійно. Зображення повністю під контролем програмних інструкцій, а не користувача. Приклад: зберігачі екрану.

Також графічні файли поділяються на методи їх створення: растрова та векторна графіки.

Растрова графіка передбачає у собі, що картинки та слова складаються з крихітних кольорових точок або квадратів, які називаються пікселями. Більшість простих комп'ютерних графічних зображень, з якими ми стикаємося, пікселюються таким чином, як стіни, збудовані з цегли. Перші комп'ютерні екрани, розроблені в середині 20-го століття, працювали багато в чому як телевізори, які раніше створювали свої зображення, що рухаються, «скануючи» пучки електронів (крихітні заряджені частинки всередині атомів, також їх називали катодними променями) вперед і назад зверху. знизу і зліва направо - як свого роду електронна кисть миттєвої дії. Такий спосіб створення зображення називається растровим скануванням, тому створення зображення на екрані комп'ютера з пікселів називається растровою графікою.

Растрові зображення використовують двійковий формат, тобто те, як комп'ютери представляють десяткові числа (1,2,3,4 і так далі), використовуючи лише дві цифри нуль і один (тому десяткове число 5678 стає 1011000101110 у двійковій комп'ютерній мові ). Припустимо, комп'ютер хоче запам'ятати картинку, яку хтось малює на екрані. Якщо малюнок чорно-білий, програма може використовувати нуль для зберігання білої області

зображення та одиницю для зберігання чорної області (або навпаки). Копіюючи кожен піксель по черзі, можна перетворити зображення, що заповнює весь екран, скажімо, 800 пікселів у діаметрі на 600 пікселів вниз, до списку з 480 000 (800x600) двійкових нулів і одиниць. Цей спосіб перетворити фотографію на комп'ютерний файл з бі ні краплі Digi тс (які називаються біти для стислості) називається бітова карта, тому що є пряма відповідність-один-до-одного «відображення» - between кожного пікселя зображення і кожен біт у файлі. Насправді більшість растрових зображень є кольорові зображення. Якщо використовуємо один біт для представлення кожного пікселя, можна лише визначити, чи увімкнено піксель (білий або чорний); якщо будемо використовувати (скажімо) вісім біт для представлення кожного пікселя, зможемо запам'ятати вісім різних кольорів, але тоді буде потрібно у вісім разів більше пам'яті (місця для зберігання всередині комп'ютера), щоб зберегти зображення того ж розміру. Чим більше кольорів хочемо зобразити, тим більше потрібно бітів.

Растрова графіка проста у використанні, і легко побачити, як програми, що її використовують, роблять свою справу. Якщо намалювати піксельне зображення на екрані комп'ютера і натиснути кнопку в графічному пакеті, щоб «відобразити» зображення (переверніть його зліва направо або праворуч наліво), все, що зробить комп'ютер, це змінить порядок пікселів, щоб поміняти місцями послідовність нулів та одиниць, які їх представляють. Якщо масштабувати зображення так, щоб воно було в два рази більше, комп'ютер двічі копіює кожен піксель (так що числа 10110 стають 1100111100), але зображення в процесі стає помітно більш зернистим і піксельним. Це один із основних недоліків використання растрової графіки: вони погано масштабуються до різних розмірів. Ще один недолік - це обсяг пам'яті, який їм потрібний. Для дійсно деталізованої фотографії може знадобитися 16 мільйонів кольорів, що передбачає зберігання 24 бітів на піксель і в 24 рази

більше пам'яті, ніж базове чорно-біле зображення. (Якщо підрахувати, буде виявлено, що зображення, які повністю заповнюють комп'ютерний монітор з роздільною здатністю 1024x768 і використовують 24 біти на піксель, вимагає приблизно 2,5 мегабайта пам'яті.).

Векторна графіка. Є альтернативний метод комп'ютерної графіки, що дозволяє оминути проблеми растрової графіки. Замість того, щоб будувати картинку з пікселів, малюємо її інакше, використовуючи прості прямі та вигнуті лінії, які називаються векторами або базовими формами (колами, кривими, трикутниками тощо), відомими як примітиви. За допомогою растрової графіки ви можете намалювати будинок, побудувавши його із сотень, тисяч або мільйонів окремих пікселів; що важливо, кожен піксель не пов'язаний з жодним іншим пікселем. За допомогою векторної графіки можна намалювати прямокутник для основного будинку, прямокутники меншого розміру для вікон і дверей, циліндр для димової труби і багатокутник для даху. Дивлячись на екран, векторний графічний будинок все ще здається намальованим пікселів, але тепер пікселі точно пов'язані один з одним - вони точки вздовж різних ліній або інших фігур, які намалювали. Малювання з прямими лініями та кривими замість окремих точок означає, що можна швидше створювати зображення та зберігати його з меншою кількістю інформації: є можливість описати намальований вектор будинку як «два червоні трикутники і червоний прямокутник (дах), що розміщені на коричневому прямокутнику (головна будівля))», але не можна так просто резюмувати піксельне зображення. Також набагато простіше масштабувати векторне зображення вгору і вниз, застосовуючи математичні формули - алгоритми, які перетворюють вектори, з яких створюється зображення. Ось як комп'ютерні програми можуть масштабувати шрифти до різних розмірів, не роблячи їх піксельними та зернистими.

Більшість сучасних пакетів комп'ютерної графіки дозволяють малювати зображення, використовуючи суміш растрової або векторної графіки, за бажанням користувача, тому що іноді один підхід працює краще, ніж інший, а іноді вам потрібно змішати обидва типи графіки в одному зображенні. За допомогою графічного пакета, такого як GIMP (програма маніпулювання зображеннями GNU), можна малювати криві на екрані, викреслюючи і потім заповнюючи «контури» (технічно відомі як криві Безьє) перед перетворенням їх на пікселі («розтеризація») включити їх у щось на зразок растрового зображення.

3D графіка. Реальне життя не схоже на комп'ютерну гру або симулятор віртуальної реальності[5]. Найкращі анімації CGI (комп'ютерні зображення) легко відрізнити від тих, які зроблені на плівці чи відео з реальними акторами. Якщо дивитися на об'єкти в навколишньому світі, здається, що вони не намальовані ні з пікселів, ні з векторів. Миттєво мозок збирає набагато більше інформації з реального світу, ніж художники можуть включити навіть у найреалістичніші комп'ютерні графічні зображення. Щоб комп'ютерне зображення виглядало так само реалістично, як фотографія (не кажучи вже про реальну сцену), нам потрібно включити набагато більше, ніж мільйони кольорових пікселів[1].

Дійсно складні програми комп'ютерної графіки використовують цілу низку методів, щоб намальовані від руки (і часто цілком уявні) двовимірні зображення виглядали принаймні так само реалістично, як фотографії. Найпростіший спосіб домогтися цього – покладатися на ті ж хитрощі, які завжди використовували художники, - такі речі, як перспектива (як об'єкти віддаляються вдалину до «точки сходу» на горизонті) і усунення прихованих поверхонь (коли довколишні предмети частково неясні, які знаходяться далі ).

Якщо потрібні реалістичні тривимірні зображення для таких речей, як САПР (комп'ютерне проектування) та віртуальна реальність, необхідно застосувати набагато складніші графічні методи. Замість малювання об'єкта, створюєте його тривимірну комп'ютерну модель усередині комп'ютера і різними способами маніпулюєте ним на екрані[12]. По-перше, створюється базовий тривимірний контур об'єкта, званий каркасом (бо він намальований з векторів, які виглядають так, ніби вони можуть бути невеликими металевими дротиками). Потім модель налаштовується - процес, у якому різні частини об'єкта пов'язані один з одним, як кістки у скелеті, тому вони рухаються разом реалістично[10]. Нарешті, об'єкт візуалізується, що включає зафарбовування зовнішніх частин за допомогою різних текстур (зразків поверхні), кольорів, ступеня непрозорості або прозорості і т. д. Рендеринг - це надзвичайно складний процес, який може зайняти у потужного комп'ютера години, дні або навіть тиждень. Складна математика використовується для моделювання того, як світло падає на поверхню, зазвичай з використанням або трасування променів (відносно простий метод побудови прямих ліній відбиття світла від поверхні блискучих об'єктів), або випромінювання (складніший метод моделювання того, як повсякденні предмети відбивають та розсіюють світло) більш тьмяними і складними способами.

## 1.2 Технічні види обробки зображень

Існують різні можливості обробки зображень задля подальшого використання цих зображень. Розгляну деякі з них.

Корекція кольору зображення:

1) усунення колірних дефектів,
2) зміна загального балансу кольорів.

Розглянемо алгоритм для усунення дефектів на рисунку 1.1. Він обробляє зображення та придає йому той колір, що має освітлення навколо, та отримуємо зображення, що пройшло корекцію (рис 1.2).

```
import cv2
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression

example = cv2.imread('cc_part12.png')
poly = PolynomialFeatures(degree=2)
X_ = poly.fit_transform(X)
clf = linear_model.LinearRegression()
t = clf.fit(X_, vector)

img = example.copy()
height, width, depth = img.shape
print height, width, img.shape
for i in range(0, height):
    for j in range(0, (width)):
        predict_ = poly.fit_transform(example[i, j])
        img[i,j] = clf.predict(predict_)[0]
cv2.imwrite('out.png', img)
```

Рисунок 1.1 — Приклад програмної обробки зображення

Рисунок 1.2 — Результат роботи

Згладжування – це відображення плавно намальованих кривих на піксельному дисплеї може призвести до появи нерівних країв («нерівностей»). Одним із рішень цього є розмиття пікселів на кривій, щоб отримати гладкішу лінію. Цей метод відомий як згладжування широко використовується для згладжування шрифтів на піксельних екранах комп'ютерів[18].

## 1.3 Галузі використання

Очевидні застосування комп'ютерної графіки включають комп'ютерне мистецтво, фільми CGI, архітектурні креслення та графічний дизайн - але є також багато неочевидних застосувань, і не всі є «художніми». Наукова візуалізація – це спосіб створення графічного виведення комп'ютерних моделей, що спрощує розуміння людьми. Комп'ютеризовані моделі глобального потепління виробляють на виході великі таблиці чисел, які може визначити тільки доктор наук у галузі клімату; але якщо створити прискорену анімовану візуалізацію - Земля має синіший колір в міру того, як стає холодніше і червоніший в міру того, як стає спекотніше, - будь-хто зможе

зрозуміти, що відбувається. Медична візуалізація - ще один хороший приклад того, як графіка робить комп'ютерні дані більш значущими. Коли лікарі показують сканування мозку або тіла, можна побачити комп'ютерне графічне зображення, побудоване з використанням величезних обсягів даних, отриманих із тисяч або навіть мільйонів вимірювань. Приголомшливі фотографії, отримані з космосу дивовижними пристроями, такими як космічний телескоп Хаббл, зазвичай покращуються за допомогою комп'ютерної графіки, яка називається обробкою зображень; це може здатися складним, але це не дуже відрізняється від використання графічного пакета, такого як PhotoShop, для коригування святкових знімків).

Можна сказати, що це справді ключовий момент у комп'ютерній графіці: вони перетворюють складну комп'ютерну науку на повсякденне мистецтво, яке можна зрозуміти миттєво та інтуїтивно. Ще в 1980-х, коли програмувався Commodore PET, єдиний спосіб був - набирати такі слова, як PEEK і POKE, на чорно-зеленому екрані. Практично кожен сучасний комп'ютер тепер має так званий GUI (графічний інтерфейс користувача), що означає, що керує машиною користувач, вказуючи на потрібні об'єкти, клацаючи їх мишею або пальцем або перетягуючи їх по своєму робочому столу. Це має набагато більше сенсу, тому що людина візуалізує: щось на зразок третини нашої кори (вищого мозку) зайняте обробкою інформації, яка надходить у нашу голову через очі. Ось чому картинка дійсно коштує тисячі слів (іноді набагато більше), і чому комп'ютери, які допомагають нам візуалізувати речі за допомогою комп'ютерної графіки, справді революціонізували те, як ми бачимо світ.

# Висновки до першого розділу

В першому розділі досліджено особливості растрової та векторної графіки. Які є специфіки роботи з різними графічними об’єктами та необхідність використання.

Також розглянуто основні правила та виключення корекцій фотокартки. Які є способи. Додано приклади програмної реалізації.

Розглянуті різні галузі використання модифікацій зображень. Для чого необхідно покращувати картинки, виділяти межі та одним словом обробляти графічні файли. Програмна обробка багатьох фотокарток здебільшого необхідна для подальшого використання у комп’ютерному зорі, що полегшує життя людини.

# 2 ОПИС МЕТОДІВ ТА АЛГОРИТМІВ ОБРОБКИ ЗОБРАЖЕННЯ ТА ВИЯВЛЕННЯ МЕЖ ОБ'ЄКТІВ.

Фотографія може бути зроблена так, що будуть видно деякі недоліки якості. Недоліки якості зображення бувають такі:

- погано налаштовано яскравість;
- фотокартка не різка;
- не чітко видно контур предметів;
- поганий фокус, якщо деякі предмети розміщені так, що один з них потрапляє в розфокусування (рис 2.1).

Рисунок 2.1 – Приклад різних фокусувань на предметах зображення в залежності від відстані один від одного

Ці проблеми зустрічаються у різних фотографіях, мікроскопії та якщо необхідно вилучити стоп-кадр з камери спостереження. У мікроскопії вчені роблять кадр з різних ракурсів та кутів, щоб отримати можливість зробити реальну структуру предмету за яким спостерігають, тобто 3D модель. Стоп-кадри з камер відеоспостереження повинні бути чіткими, щоб можливо було розпізнати лице злодія, номер машини тощо. Також існують старі фотографії, в які можна вдихнути нове життя (зробити більш чітким, сфокусованим та налаштувати колір).

На сьогоднішній час це можливо реалізувати за допомогою декількох алгоритмів та методів. Ці алгоритми виконують різні функції. Наприклад, знаходження контурів зображення та корекція фотокартки в цілому. Деякі методи включають в собі поєднання цих функцій та працюють за допомогою поєднання інших.

Спробуємо в даному проєкті розглянути існуючі методи рішення для обробки зображення та алгоритми, які використовуються.

## 2.1 Існуючі методи рішення для обробки зображення

### 2.1.1 Алгоритм мультифокуса

Алгоритм мультифокуса частіше за все застосовується для обробки графічних файлів, котрі було здобуто за допомогою мікроскопів. Для побудови зображення за допомогою алгоритма необхідно зробити декілька фотокарток с фокусуванням на різних предметах[7].

Алгоритм мультифокуса працює з декількома кадрами та поєднує їх в один за допомогою піксельних даних.

Етапи роботи алгоритму:

1) приймаються серії фотокарток з різним фокусуванням (тобто створюється серія фотокарток);
2) зображення розміщуються в одну площину;
3) поділяються на зони (окремо взяті ділянки фотокарток);
4) порівнюються зони з різних фотокарток цієї серії;
5) виділяється найкраща зона;
6) в результаті отримується зображення, яке складає найкращі зони фотографії (рис. 2.2).

Рисунок 2.2 - Серія фотокарток с різним фокусуванням

Переваги:

1) отримані зображення мають меншу кількість артефактів ніж алгоритми, що були до алгоритма мультифокуса.

Недоліки:

1) не чітке виділення дільниці де повинна бути присутня різкість в зображенні (рис. 2.2);

2) може бути присутнє змішування кольорів (рис. 2.3);

3) може бути присутнє змінення масштабу ділянки (рис. 2.4).

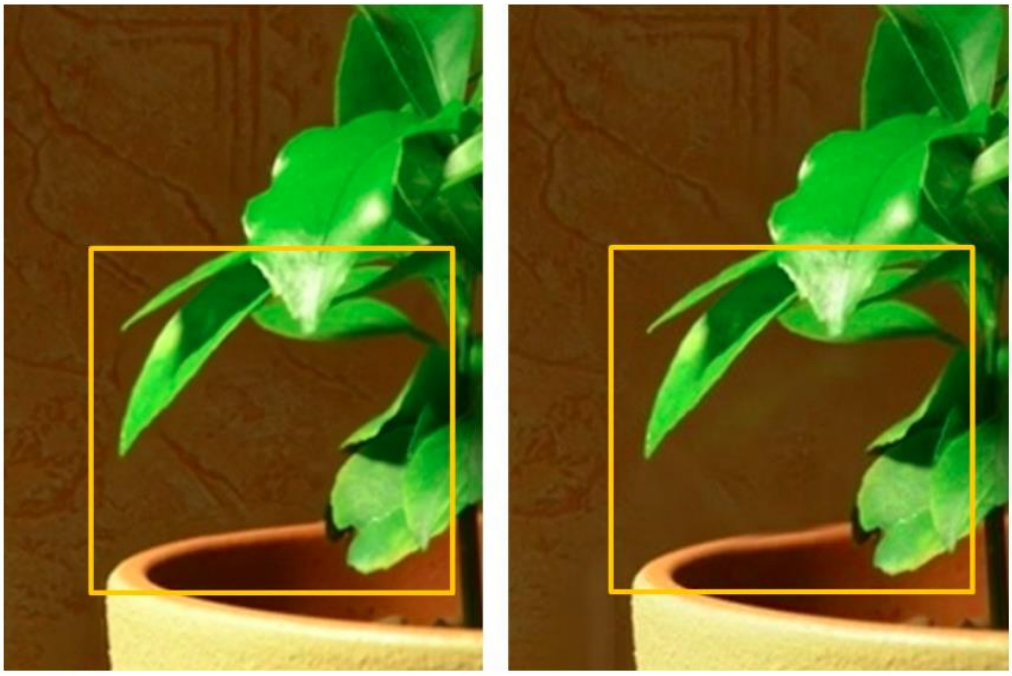


Рисунок 2.2 – Ділянка в розфокусі

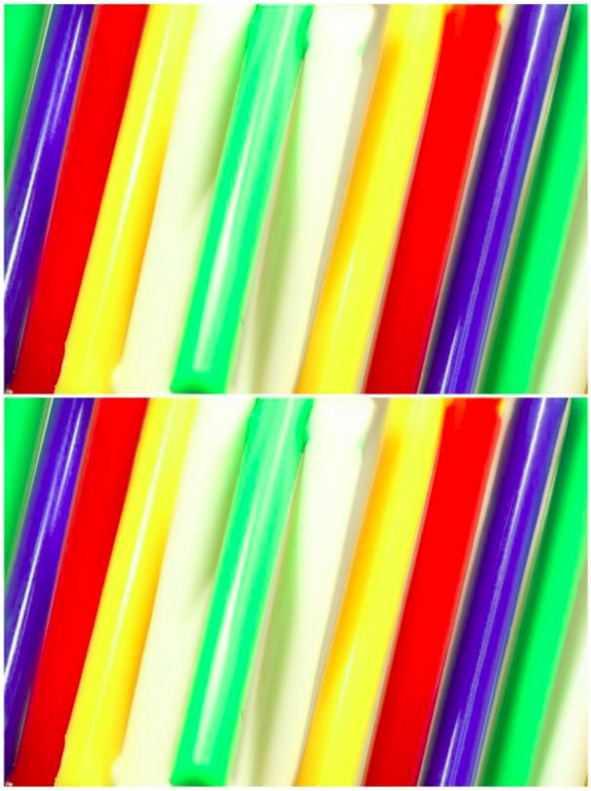


Рисунок 2.3 – Змішування кольорів

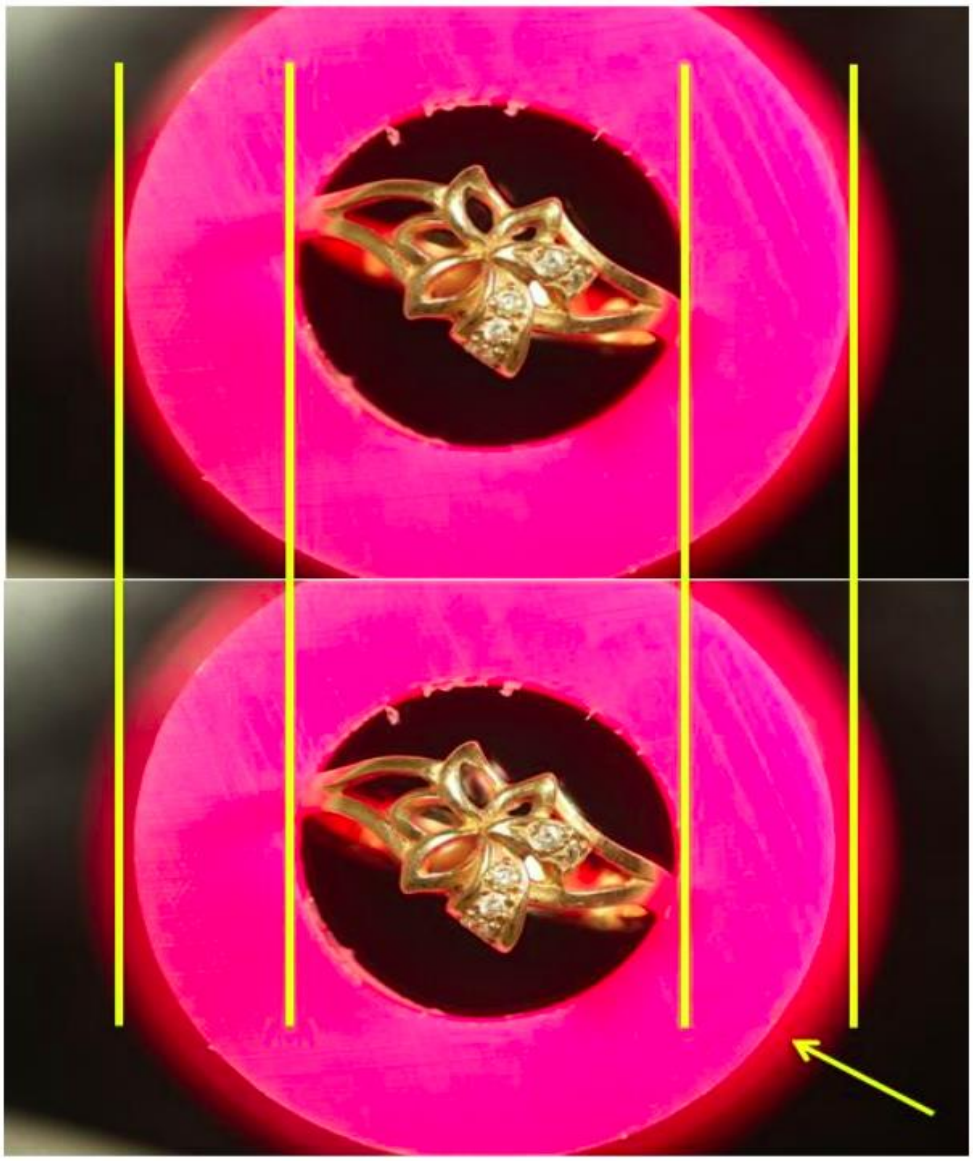


Рисунок 2.4 – Змінення масштабу предмета

### 2.1.2 Детектор Моравеца

Детектор Моравеца працює з використанням особливих точок.

Принцип роботи алгоритму такий:

1) виділяється квадратне вікно зображення W (розміром 3X3, 5X5, 7X7);
2) розглядається зміна яскравості вікон;
3) використовується алгоритм зсуву точки, котра нас цікавить:

a) для усіх пікселей з координатами *(x, y)* вирахувати зміну інтенсивності.

$$V_{u,v}(x,y) = \sum_{\forall a,b \epsilon w} (I(x+u+a, y+v+b) - I(x+a, y+b))^2, (1)$$

де $(u,v) \epsilon \{(1;0),(1;1), (0;1),(-1;1),(-1;0),(-1;1),(0,-1),(1;-1)\}$

b) Визначити направлення де є найменша інтенсивність зміни пікселів (де пікселі схожі один на одного);

c) відняти між собою пікселі, які менше значення *T;*

d) видалити повторювані вугли ребер. Для цього необхідно визначити локальні максимуми функції.

Усі ненульові елементи будуть відповідати вимогам зображення, котре шукали.(рис 2.4).

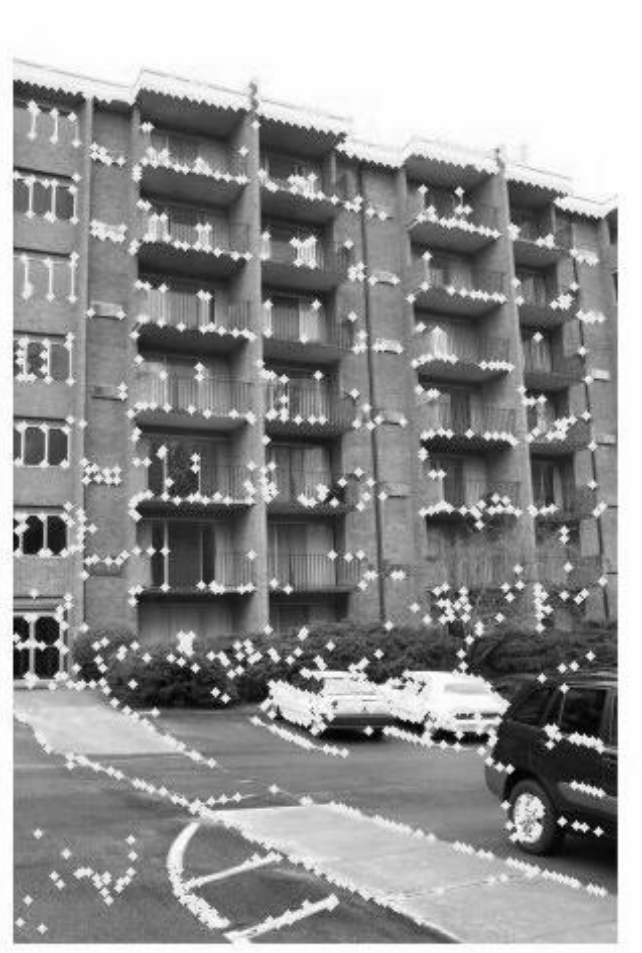


Рисунок 2.4 - Виділення вуглів ребер за детектором Моревеца

Переваги:

1) простий у використанні;

Недоліки:

1) відсутність інваріантності щодо повороту предмета на зображенні;

2) якщо є велика кількість ребер на зображенні, то не чітко відображає перехід.

### 2.1.3 Алгоритм Лукаса-Канаде

Алгоритм Лукаса — Канаде — широко використовуваний у комп'ютерному зорі диференційний локальний метод обчислення оптичного потоку (рис 2.5).

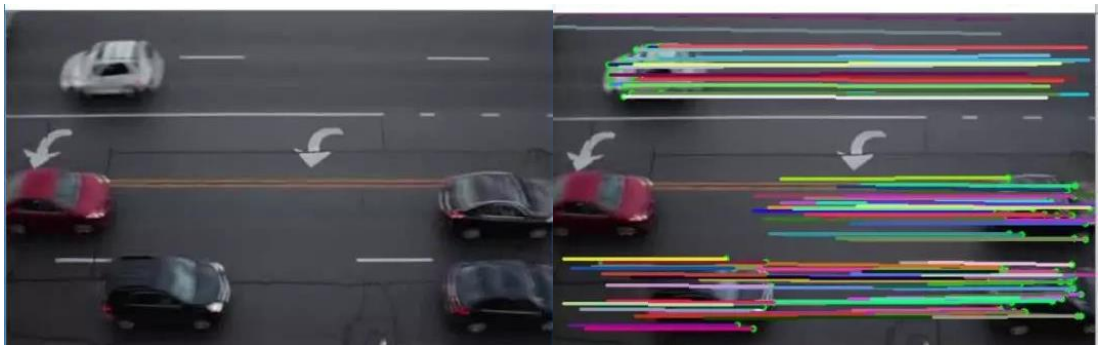


Рисунок 2.5 – Оптичний потік

Основне рівняння оптичного потоку містить дві незалежні змінні і не може бути однозначно розв'язаним[11]. Алгоритм Лукаса-Канаде вирішує неоднозначність за рахунок використання інформації про сусідні пікселі в кожній точці. Метод ґрунтується на припущенні, що в локальному околі кожного пікселя значення оптичного потоку однакове, таким чином можна записати рівняння оптичного потоку для всіх пікселів в околі і розв'язати систему рівнянь методом найменших квадратів.

Оптичний потік містить важливу інформацію про структуру сцени.

Для кожного пікселя одного кадру необхідно обчислити вектор зміщення, зсув пікселя від поточного кадру до наступного. Це завдання схоже на щільне зіставлення: для кожного пікселя одного кадру знайти ту ж точку на іншому кадрі.

Можна визначити область руху, приписавши кожній точці зображення вектор швидкості. В деякий момент часу точка Pi на зображенні відповідає певній точці *P0*. Ці дві точки пов'язані між собою рівняннями проектування. Точка *P0* переміщується щодо спостерігача (камери, очей) зі швидкістю *v0*.

Операторною проблемою є неоднозначність зміщення при обмеженому полі зору для періодичних зображень. Наприклад, коли в поле зору потрапляє фрагмент зображення, в якому присутня деяка циклічність. Внаслідок шумів в таких неоднозначних ситуаціях буде отримано не нульовий визначник, а надто маленький, який призведе до більших значень зсуву, що буде мало схоже на дійсність[10].

Метод не ефективний в тому випадку, коли припущення невірні. Наприклад, колір є непостійним, існує рух між кадрами, відбувається рух сусідніх пікселів відповідно вибраного.

Для вирішення цих проблем можна скористатися поліпшеними варіантами методу Лукаса-Канаде.

Якщо не вдається обчислити оптичний потік за одну ітерацію, то можна застосовувати ітеративний метод. Суть методу полягає в наступному: є одновимірні функції f1 (x) на першому кадрі і f2 (x) - на другому. Необхідно знайти вектор зміщення d.

### 2.1.4 Порегіональний метод

Ідея порегіонального метода базується на алгоритмі мультифокуса, а саме працює над недоліками алгоритма[20]. Алгоритм мультифокуса розробляє одну багатофокусну картинку через склеювання найкращих полів інших картинок. Тобто кожна картинка накладається поверх одна одної та с цих зображень обираються поля з найкращим фокусом. Даний алгоритм представлений для оптичних мікроскопів та портативних фотоапаратів.

Порегіональний метод прибрав такі недоліки алгоритму мультифокуса:

1) покращив підхід для вибору функцій, що призводить до згладжування переходів від одного регіону до іншого;

2) зробив декілька фотографій з різних кутів, що дозволяє зобразити коректну глибину зображення;

3) використав властивість інкрементації, що додало методу продуктивність.

Порегіональний метод вирішує такі задачі:

1) зменшує зображення до максимально можливого розміру;

2) виділяє особливі точки на цих зображеннях;

3) ставить точки у відповідність до кожного кадру.

Порегіональний метод та методи, як на нього схожі мають одну проблему - це вразливість до масштабування зображення та рух предметів на картинці, що призводить до спотворення ракурсу та до геометричних спотворень зображення. Отже, це необхідно обійти. Для того щоб обійти спотворення потрібно використати виділення особливих точок. Дескриптори дають змогу виділити на картинці особливі точки, однак, у дескрипторів є свої вимоги:

1) інваріантність - опис однієї і тій самій області чи точки, що лежать на різних зображеннях і не мають відрізнятись;

2) унікальність - дескриптори мають бути унікальні для різних особливостей зображення;

3) стійкість - незмінність точок коли з’являються геометричні перетворення зображення.

Вирішення задач порегіонального методу:

Для того щоб усі задачі порегіонального методу були вирішенні спершу відбувається формування дескрипторів.

Нехай, *M* - перше чорно-білу зображення, а *S* - друге, котрі формуються у час t та t+1 відповідно. Яскравість зображення має величини з координатами.

$$u = [x\ y]^T \text{- } M(u) = M(x, y) \quad (2)$$

Зображення - це дискретна величина чи масив, координати верхнього кута якого можна позначити як - $[0\ 0]^T$.Висоту зображення приймемо за *sizey*, а ширину за *sizex*. Отже нижній правий кут матиме координати - $[sizex - 1\ sizey - 1]^T$.

Розглянемо на першому зображенні *M* довільну точку $u = [u_x\ u_y]^T$, це необхідно для того щоб на іншому зображенні S взяти таку точку *v*, що:

$$v = u + d = [u_x + d_x u_y + d_y]^T \quad (3)$$

Точки *M(u)* і *S(v)* повинні бути схожі. Приймемо швидкість зображення в точці u за такий вектор $d = [d_x\ d_y]^T$.Схожість зображення буде характеризуватися тільки в деяких місцях. Це відбувається через особливості оптичної лінзи. Точки $w_x$ і $w_y$будемо приймати як два цілих числа. Виходячи з цих тверджень швидкість зображень *d* можна взяти як вектор функції δ та він буде визначитися як:

$$\delta(d) = \delta(dx, dy) = \sum_{x=u_x-w_x}^{u_x+w_x} \sum_{x=u_x-w_x}^{u_x+w_x} (\mathrm{M}(x, \mathrm{y}) - S(x + d_x, y + d_y))^2 \quad (4)$$

Зображення будуть подібні один до одного в точці, або біля точки $(2w_x + 1) * (2w_y + 1)$. Для розрахування необхідно використати детектор

Харріса(3), щоб вивести ознаки зображення та для визначення відгука кута. Обчислюється матриця:

$$M = \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \quad (5)$$

Можна спостерігати покращення або погіршення якості картинки навіть коли точки (*x, y*) змінюють свою позицію на невелику відстань. Це спостерігається, якщо власні значення матриці значні. Записується функція відгука кута таким чином:

$$R = detM - k(traceM)^2 \quad (6)$$

Максимум екстремуму функції буде знаходитись в точці *k* та дорівнює 0.4.

Точка буде вважатися особливою в тому випадку. коли вона в матриці буде більша ніж поріг в цій точці.

Якщо є матриця A та вона має розмір 2×2, а вектор зміщення в цій матриці буде 2×1, то рух пікселів на зображенні дорівнюватиме:

$$v=Au+d. \quad (7)$$

Для того щоб визначити параметр руху та спотворення зображення, необхідно стежити за пікселями на всьому малюнку. Це проблематично через надто великий об’єм цих точок, а також забирає багато пам’яті, що призводить до уповільнення роботи програми. Отже так мінімізуються різниця руху пікселів[6]:

$$\delta(d) = \int\int_w [\mathrm{I}(u) - J(A_u + d)]^2 \, w(u)du \quad (8)$$

У цій формулі є значення *w(u)* - це функція по вікну(2)

З виразу, котрий отримали необхідно взяти похідну відносно руху та прирівняти отримане до 0. Потім застосовується метод представлення

замкнутих нелінійних систем(2), це можна зробити за допомогою розкладу функції зображення в ряд Тейлора, отримуємо необхідний вираз функції:

$$J(Au + d) = J(u) + g^T(u) \quad (9)$$

Таким чином можна отримати вектор похибки зображення:

$$a = \iint_w [I(u) - J(u)] \begin{bmatrix} xg_x \\ xg_y \\ yg_x \\ yg_y \\ g_x \\ g_y \end{bmatrix} wdu \quad (10)$$

Рух точок може бути аффіним, або просто зміщенням координат. Для того щоб слідкувати за зміщенням є деякі алгоритми, котрі цим займаються, а саме:

- Lucas-Kanade - особливість вважається тільки зміщення, без спотворень;
- Tomasi-Kanade - переформулювання Lucas-Kanade. Вважається зміщенням, і розраховується шляхом ітеративного вирішення побудованої системи лінійних рівнянь;
- Shi-Tomasi-Kanade - враховує афінні спотворення особливості руху;
- Jin-Favaro-Soatto - модифікація Shi-Tomasi-Kanade з урахуванням афінних змін освітленості особливості.

Переваги:

1) працює швидше за алгоритм мультифокуса;

2) при удосконалені може бути краще зроблена обробка зображення без спотворення;

3) більш продуктивний за алгоритм мультифокуса.

Недоліки:

1) порегіональний алгоритм витрачає більше пам’яті за алгоритм мультифокуса.

2) має ті ж недоліки тільки у менших масштабах;

3) можемо побачити ареоли після обробки зображення даним методом (Рисунок 2.6);

Рисунок 2.6 – Зображення з ареолами

### 2.1.5 Алгоритм Канні

Задача алгоритму:

1. Покращення визначення координат;
2. Виявлення шумів;
3. Відгук тільки на ті координати, яку необхідно коригувати.

Функції, які використовує алгоритм:

1) для видалення шумів використовується згладжування зображення з використанням фільтра Гауса[5];

2) виділення границь зображення. Визначається найбільший перехід градієнта з одного в інший колір;

3) видалення не максимумів. Використовуються тільки ті дільниці, в яких є виявленні найчіткіші межі контурів зображення;

4) ділянки які могли б бути визначенні як межі контурів, виділяються порогамою;

5) підсумкові межі контуру, які не були визначенні як сильні границі зображення, визначаються шляхом придушення всіх інших країв.

Для найбільш коректного застосування алгоритму до зображення додають відтінки сірого для більш чіткого виділення меж контурів та зменшення витрат на обчислення (Рисунок 2.5).

Рішення задач алгоритму Канні.

Перед застосуванням методу для зменшення обчислювальних витрат вихідне зображення піддається попередній обробці. Спочатку малюнок, представлений в колірній моделі RGB, перетворять в модель *YUV* (або *HSL*, *HSV* і т.), в якій колір представляється як три компоненти - яскравість (*Y*) і дві що містять різні кольори (*U* і *V*). Для цього кожен з складових кольору множиться на коефіцієнти переведення, є постійними в зв'язку з особливостями людського сприйняття.

Для того, щоб обчислення не були дуже громіздкими перед тим, як застосовувати алгоритм Канні, зображенню необхідна попередня обробка, а саме видалення кольору з графічного об’єкта.

Якщо модель зображення RGB (дана модель має за собою властивість змішування зеленого, червоного та синього кольору завдяки чому з’являються нові кольори), то вона перетворюються на HSV (дана модель котра за собою має такі границі: Hue - колірний тон, Saturation - насиченість, *Value* (значення кольору) або *Brightness* - яскравість.) чи на *YUV* (яскравість (Y) і два кольори, котрі відрізняються компонента (U і V) [2]) перетворення в RGB:

$$Y = K_R * R(1 - K_R - K_B)G + K_B B \quad (11)$$

$$U = B - Y \quad (12)$$

$$V = R - Y \quad (13)$$

Підставимо в цю систему рівнянь коефіцієнти, котрі постійні для людського зору:

$$\begin{cases} V = 0{,}299 * R + 0{,}587 * G + 0{,}114 * B; \\ U = -0{,}14713 * R - 0{,}28886 * G + 0{,}436 * B; \\ V = 0{,}615 * R - 0{,}51499 * G - 0{,}10001 * B. \end{cases} \quad (14)$$

Якщо вирішити цю систему, то можна отримати наступну векторну функцію, так як колірна складова не належить обробці.

$$RGBtoYUV = \begin{pmatrix} 0{,}299 \\ 0{,}587 \\ 0{,}114 \end{pmatrix} \quad (15)$$

Де R, G і B - це компоненти (канали) колірної моделі RGB. Для подальшої обробки важлива тільки Y-складова, яка відповідає за яскравість [3].

Представлю результати роботи алгоритму Канні. Зліва картинка “до”. Справа “після”(рис. 2.7)

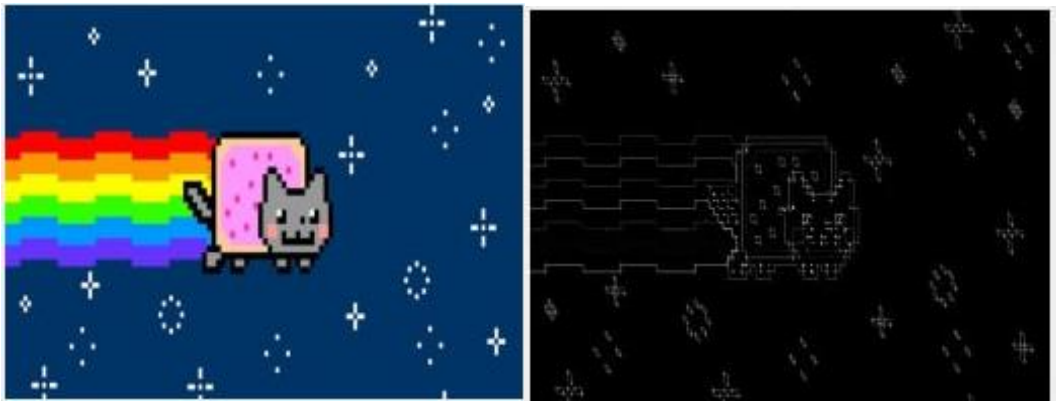


Рисунок 2.7 - Результати роботи алгоритму Канні

За підготовкою зображення, метод знаходження меж Канні застосовується наступні кроки:

- згладжування;
- відбір градієнтів;
- утримання «помилкових» максимумів;
- двоїста порогова фільтрація;
- визначення ділянки неоднозначності

Переваги:

1) швидко працює;

2) легкий для застосування;

3) можна обробити багато зображень та з різною площею;

Недоліки:

1) якщо фон має колір як і об'єкт але з різними відтінками, то контури будуть зазначені не точно.

### 2.1.6 Оператор Собеля

Оператор Собеля так само як і алгоритм Канні працює з контурами зображення. Зазвичай оператор Собеля працює з чорно-білими зображеннями. Так як його принцип роботи підходить більше для таких зображень та тих, які мають геометричні предмети (куб, круг, піраміда, тощо.)( Рисунок 2.7).

Оператор Собеля - дискретний диференціальний оператор. Призначений для пошуку границь об’єктів на зображенні. Працює на функціях порівняння контраста переходу пікселів (тобто де більший контраст там і є межа об’єктів) оператор Собеля обчислює похідну різного порядку[4].

Опис роботи оператора Собеля:

Вимірює градієнт контурів на зображенні та виділяє саме ті місця в яких параметр переходу має найбільше значення. Ці виділені точки, де параметр градієнта найбільший, і є межами контуру.

Принцип роботи заснований на згортуванні зображення, в якого є цілочисельні фільтри. Якщо взяти для прикладу найпростіший випадок, то можна побачити, що на сам перед оператор Собеля обчислює згорток вихідного зображення з ядрами, які дорівнюють $G_x$ та $G_y$, для прикладу обчислення похідних візьмемо такі напрямки:

$$G_I=\begin{bmatrix}-1 & 0 & 1\\ -2 & 0 & 2\\ -1 & 0 & 1\end{bmatrix} \quad (16)$$

$$G_y=\begin{bmatrix}-1 & -2 & -1\\ 0 & 0 & 0\\ 1 & 2 & 1\end{bmatrix} \quad (17)$$

Обчислення градації кольору пікселів використовується як раз оператор Собеля. Щоб дізнатися значення першої похідної зміни інтенсивності в горизонтальному напрямку необхідно використати оператор $G_x$ , $G_y$ - у вертикальному. За допомогою обчислених операторів можна обчислити градієнт для пікселя з координатами *(i, j)* . Він буде дорівнювати:

$$|G^{ij}| = \sqrt{(G_x^{ij})^2 + (G_y^{ij})^2} \quad (18)$$

Використовуючи оператори також можливо визначити напрямок градієнта:

$$\theta^{ij} = \mathrm{a}rctan(\frac{G_y}{G_y}) \quad (19)$$

У бібліотеці OpenCV підтримується обчислення перших, других, третіх і змішаних похідних функції інтенсивності пікселів з використанням розширеного оператора Собеля. Нижче наведено прототип відповідної програмної функції[26].

*void Sobel_operator (const M & src, M & dst, int wdepth, int Ox, int Oy, int nsize =3, double scale = 1, double dd = 0, int border = BORDER_DEFAULT)*

Параметри функції *Sobel_operator:*

- *src* - вхідне зображення, котре необхідно обробити;
- *dst* - вихідне зображення, те що отримали.
- *wdepth* - глибина результуючого зображення.
- *Ox* - порядок похідної по осі Ox.
- *Oy* - порядок похідної по осі Oy.

- *nsize* - розмір розширеного ядра оператора Собеля. Розмір розширеного ядра (nsize) може отримати одне значення (1, 3, 5, 7). Розмір ядра буде nsize x nsize завжди. Окрім випадку, коли nsize = 1. Коли розмір дорівнює один, то розмір ядра буде 1x3 чи 3x, для того щоб мати змогу застосовувати фільтр Гаусса. За допомогою обраного значення (1, 3, 5 або 7) використовується для обчислення першої та другої приватної похідної по осях Ox та Oy. Стандартно для обчислення використовується розмір 3x3. Також можна виділити те, що в алгоритмі передбачено значення параметру nsize, котре буде рівне CV_SCHARR = -1. Коли використовується -1 розмір ядра автоматично стає 3x3 і через це можна давати більш точні оцінки похідних в порівнянні з оператором Собеля. Так як при цьому розмірі алгоритм використовує фільтр Щарри.

$$G_x = \begin{bmatrix} -3 & 0 & 3 \\ -10 & 0 & 10 \\ -3 & 0 & 3 \end{bmatrix}, G_y = \begin{bmatrix} -3 & -10 & -3 \\ 0 & 0 & 0 \\ 3 & 10 & 3 \end{bmatrix} \quad (18)$$

- *scale* - це опційний параметр. Він використовується для того щоб можна було задати коефіцієнт масштабування. Коефіцієнт масштабування необхідний для обчислення похідних. Параметру за замовченням немає. Користувач має сам його вибрати;
- *dd* - це також опційний параметр для зміщення інтенсивності. Він додається перед збереженням результатів в матрицю dst;
- *border* - це параметр, котрий визначає метод доповнення кордону.

Приведу приклад виділення кордонів на малюнку завдяки застосуванню вертикальних і горизонтальних операторів Собеля. За допомогою усереднення градієнтів за напрямками. Також треба приділити увагу тому факту, що в алгоритмі спершу використовується фільтр Гаусса, щоб видалити шуми на вхідному графічному файлі та на зображення падає фільтр сірого. Це необхідно, щоб оператор Собеля працював без перешкод. Також можна зауважити, що на вихідному об’єкті буде відрізнятись глибина зображення.

```
#include <stdio.h>
#include <opencv2/opencv.hpp>
using namespace cv;
const char helper[] =
  "Sample_Sobel.exe <img_file>\n\
  \t<img_file> - image file name\n";

int main(int argc, char* argv[])
{
  const char *initialWinName = "Initial Image",
    *xGradWinName = "Gradient in the direction Ox",
    *yGradWinName = "Gradient in the direction Oy",
    *gradWinName = "Gradient";
  int wdepth = CV_16S;
  double alpha = 0.5, beta = 0.5;
  Mat img, grayImg, xGrad, yGrad,
    xGradAbs, yGradAbs, grad;
  if (argc < 2)
  {
    printf("%s", helper);
    return 1;
  }
// Завантаження зображення
  img = imread(argv[1], 1);
  // згладжування за допомогою фільтра Гаусса
  GaussianBlur(img, img, Size(3,3),
      0, 0, BORDER_DEFAULT);
```

```
// перетворення на відтінки сірого
  cvtColor(img, grayImg, CV_RGB2GRAY);
// Обчислення похідних за двома напрямками:
  Sobel_operator(grayImg, xGrad, ddepth, 1, 0); // по Ox
  Sobel_operator(grayImg, yGrad, ddepth, 0, 1); // по Oy
// перетворення градієнтів на 8-бітні
  convertScaleAbs(xGrad, xGradAbs);
  convertScaleAbs(yGrad, yGradAbs);
// поелементне обчислення виваженої
// знаходження суми 2 масивів
  addWeighted(xGradAbs, alpha, yGradAbs, beta, 0, grad);
// відображення результату
  namedWindow(initialWinName, CV_WINDOW_AUTOSIZE);
  namedWindow(xGradWinName, CV_WINDOW_AUTOSIZE);
  namedWindow(yGradWinName, CV_WINDOW_AUTOSIZE);
  namedWindow(gradWinName, CV_WINDOW_AUTOSIZE);
  mshow(initialWinName, img);
  imshow(xGradWinName, xGradAbs);
  imshow(yGradWinName, yGradAbs);
  imshow(gradWinName, grad);
// завершення програми
 waitKey();
// Закриття вікон
  destroyAllWindows();
// Звільнення пам'яті
  img.release();
  grayImg.release();
```

```
    xGrad.release();
    yGrad.release();
    xGradAbs.release();
    yGradAbs.release();
    return 0;
}
```

Результати роботи програми приведено на рисунку (рис 2.8): зліва картинка “до”, справа - “після”.

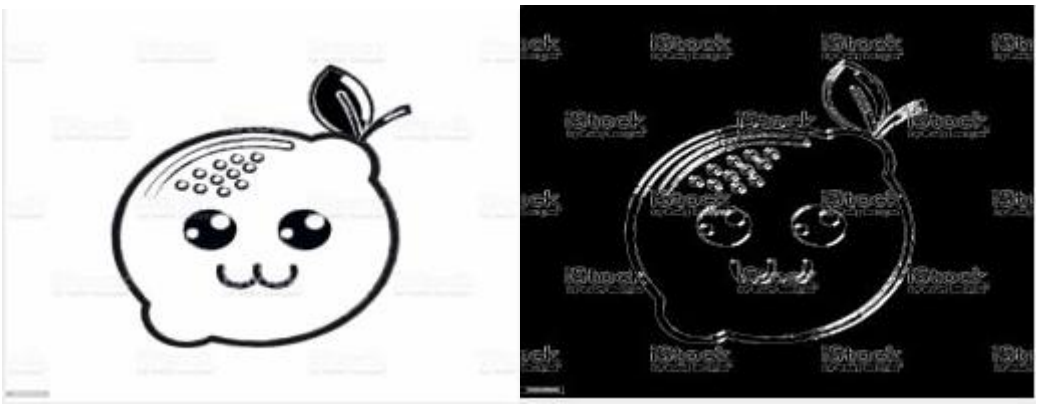


Рисунок 2.8 - Результати роботи оператора Собеля

Переваги:

- згладжує краще ніж алгоритм Канні;
- ділянки виділяє краще;
- легкий для застосування та розуміння.

Недоліки:

- працює повільніше ніж алгоритм Канні;
- відрізняється глибина зображення;
- необхідно щоб зображення було чорно-біле.

### 2.1.7 Оператор Превітта

Оператор Превітта - це один з методів, що має за собою функцію виділення кордонів об’єктів на зображенні при обробці графічного файлу. Метод був розроблений доктором Джудіт Прюїтт (Judith Prewitt) для обробки медичних зображень. Оператор обчислює максимальний відгук на безлічі ядер згортки для знаходження локальної орієнтації кордону в кожному пікселі[15].

Друга назва методу - edge template matching (укр. підстановка шаблонів кордонів), так як кожному файлу зіставляється набір шаблонів, і кожен представляє собою різну орієнтацію межі. Величину та орієнтацію границі об’єкта в пікселях визначається також шаблоном, що відповідає околиці пікселя.

Величина і орієнтація границі об’єкта визначається за допомогою детектора Превітта. У той час як детектор з диференціальним градієнтом потребує трудомісткого обчислення оцінки орієнтації за величинами у вертикальному і горизонтальному напрямках, детектор кордонів Превітта дає напрям прямо з ядра з максимальним результатом. Набір ядер обмежений 8 можливими напрямами, більшість прямих оцінок орієнтації теж дуже точні[10].

Кожен набір ядер потребує 8 згорток для кожного пікселя, а якщо брати градієнтний метод, то він потребує лише 2 згортування чутливих по вертикалі та по горизонталі.

У методі використовується два ядра 3×3, згортаючи вихідне зображення для обчислення наближених похідних значень: одне по вертикалі і одне по горизонталі. Вихідне зображення буде $G_x$і $G_y$

Оператор використовує два ядра 3×3, згортаючи вихідне зображення для обчислення наближених похідних значень: одне по горизонталі і одне по вертикалі. Покладемо вихідним зображенням, і Gx, Gy - двома зображеннями,

в яких кожна точка містить вертикальне і горизонтальне наближення похідної, яка розраховується як:

$$\mathbf{G_x} = \begin{bmatrix} -1 & 0 & +1 \\ -1 & 0 & +1 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G_y} = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ +1 & +1 & +1 \end{bmatrix} * \mathbf{A} \tag{20}$$

### 2.1.8 Оператор Кірша

Цей метод розробив Кірш у 1971 році. Алгоритм базується на одній масці, котру можна обернути по головним сторонам світу: захід, південний захід, південь, північ, північний захід, південний схід, схід та північний схід. Матриця цих масок виглядає так:

$$H1 = \begin{matrix} 5 & 5 & -5 \\ -3 & 0 & -3 \\ -3 & -3 & -3 \end{matrix}$$

$$H2 = \begin{matrix} 5 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & -3 & -3 \end{matrix} \tag{21}$$

Максимальне значення, котре було виявлено за допомогою маски - це і є розмір кордону. Максимальна величина йде за напрямком. Наприклад, маска $k_5$відповідає діагональній межі, а $k_0$- вертикальній. Останні чотири макси мало відрізняються від перших. Тобто, вони є дзеркальним відображенням щодо центральної осі матриці.

### 2.1.9 Оператор Лапласа

Оператор Лапласа - диференціальний оператор, що діє в лінійному просторі гладких функцій. Він є найпростішим ізотропним диференціальним оператором і має обертальну інваріантність.[17] Перетворення Лапласа двовимірної функції зображення є ізотропною другою похідною, що визначається як:

$$L(f) = \frac{\partial^2 f}{\partial^2 x^2} + \frac{\partial^2 f}{\partial^2 y^2} \quad (22)$$

Роботу та логіку алгоритму можна розібрати за допомогою графіків.

Якщо є функція *f*(*t*) (рис 2.9) - лінійна функція. Та коли взяти похідну *f*`(*t*) (рис 2.10) в точці *t*, то можна побачити помітний стрибок графіка в цій точці. Тобто пік функції (або якщо брати с зору на малюнок стрибок яскравості рисунку в околі точці, що досліджується).

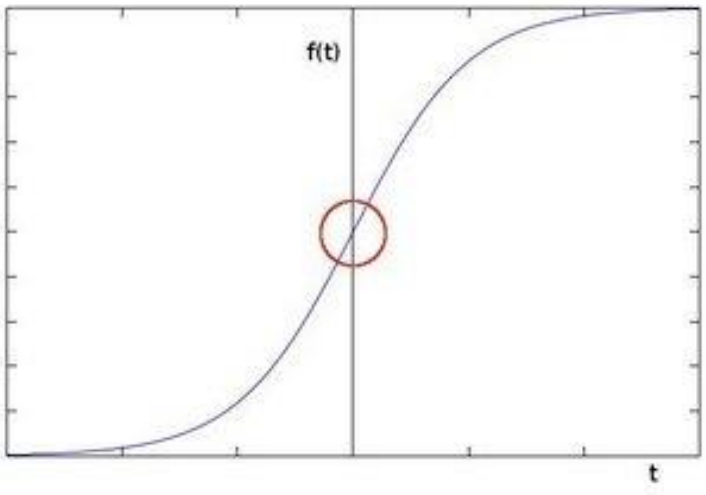



Рисунок 2.9 – Лінійна функція *f*(*t*)

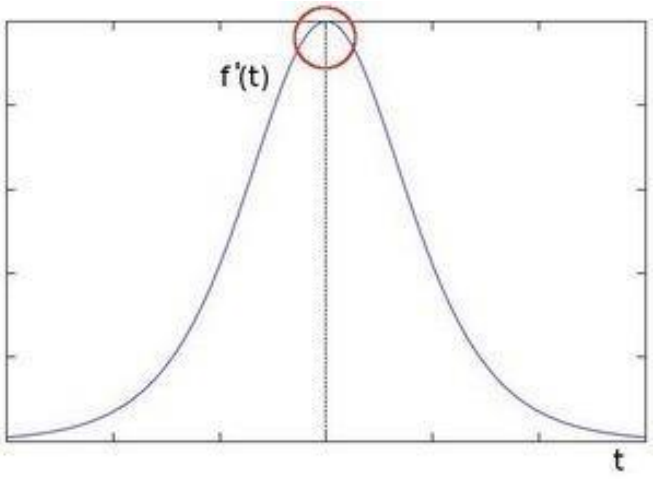



Рисунок 2.10 – Похідна функції *f*(*t*)

А якщо взяти ще одну похідну, то отримаємо *f*``(*t*) (рис. 1.11) - подвійна похідна в заданій точці.

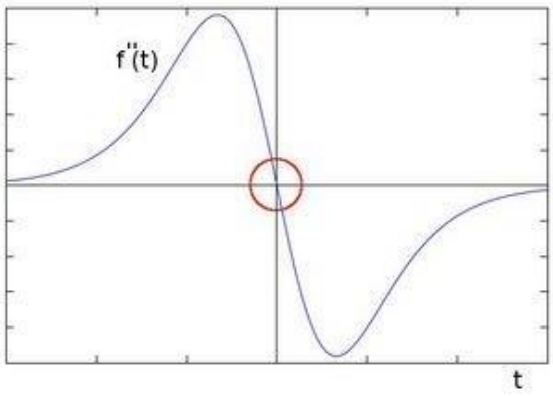



Рисунок 2.11 – Подвійна похідна функції *f*(*t*)

На графіку, що демонструє подвійну похідну функції, можна побачити, що значення точки дорівнює нулю. Отже, це можна використовувати в якості прийому для виявлення країв зображення. Є один недолік, значення точки в подвійній функції може бути нульовим не тільки на границі предмету, а й в деяких інших місцях, де не потрібно. Щоб це уникнути необхідно відфільтрувати ці точки.

Щоб функції, що наведені вище, краще обробляти зображення на програмній частині, то їх можна призвести оператором Лапласа до дискретної форми:

$$\nabla^2 f = [f(x+1,y) + f(x-1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1)] - f(x,y) \quad (23)$$

Так як оператор Лапласа працює з чорно-білими зображеннями, то можна відзначити, що обробляються відтінки які мають меншу та більшу яскравість. Це можна зобразити в вигляді матриці (рис 2.12). В таблиці зліва до обробки оператором Лапласа, справа – після

| | | |
|---|---|---|
| 0 | 1 | 0 |
| 1 | -4 | 1 |
| 0 | 1 | 0 |

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | -8 | 1 |
| 1 | 1 | 1 |

Рисунок 2.12 — Матриця обробки яскравих відтінків

| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 4 | -1 |
| 0 | -1 | 0 |

| | | |
|---|---|---|
| -1 | 1 | -1 |
| 1 | 8 | -1 |
| -1 | 1 | -1 |

Рисунок 2.13 — Матриця обробки темних відтінків

За допомогою цієї матриці можна побачити, що оператор Лапласа яскравим відтінком, котрий знаходиться між темними, стає ще яскравішим, а темний, котрий розміщений серед яскравих буде темніший, що допомагає виділяти чорний контур.

Коли є деякі шуми на малюнку, кольори на зображенні можуть бути розмиті. Отже, спершу перед визначенням кордонів об’єктів їх необхідно

згладити та забрати шуми, котрі можуть впливати на якість обробки графічного об’єкта.

Функція, котра буде підвищувати різкість зображення та забирати шуми працює так: підвищує контраст сірого рівня, отже, покращується чіткість малюнку. Фотокартка піддається усередненню, що допомагає виділяти головні деталі на зображені. Отже фотокартка стає кращою. Основний метод заточування та надання різкості в операторі Лапласа можна виразити такою формулою:

$$g(x, y = \begin{cases} f(x,y) - \nabla^2 f(x,y) \\ f(x,y) + \nabla^2 f(x,y) \end{cases} \quad (24)$$

При використанні цього кроку можна забрати шум та придати різкості, що допоможе зберегти фон на малюнку. Тобто, не тільки основні об’єкти а й фон.

Приклад використання оператору Лапласа(рис 2.14)

```cpp
#include <opencv2/opencv.hpp>
#include <iostream>
using namespace cv;

int main(int, char** argv)
{
    Mat src, gblur_src, gray_src, laplace_src, dst;
    // 1. Завантажуємо вихідне зображення src
    src = imread("images/02.png");
    if (src.empty()) { // Обнаружить картинку
        printf("could not load image...");
        return -1;
    }
    imshow("input", src);// Вывод изображения

    // 2. Спочатку розмиття, щоб забрати шум.
    //Застосуйте cv :: GaussianBlur до нашого зображення,
    // щоб зменшити шум 4(розмір ядра = 3)
    GaussianBlur(src, gblur_src, Size(3, 3), 0, 0, BORDER_DEFAULT);
    // 3. Перетворіть відфільтроване зображення
    //на зображення у відтінках сірого:
    cvtColor(gblur_src, gray_src, COLOR_RGB2GRAY);

    // 4. Використовуйте оператор Лапланса для маргіналізації зображення
    Laplacian(gray_src, laplace_src, CV_16S, 3);
    // Значение параметра
    //согласуется с приведенной выше функцией Sobel Scharr
    convertScaleAbs(laplace_src, laplace_src);
    imshow("Laplance Demo", laplace_src);

    threshold(laplace_src, dst, 0, 255, THRESH_OTSU | THRESH_BINARY);
    // Бінарізація, відображення країв чіткіше
    imshow("dst", dst);

    waitKey(0);
    return 0;
}
```

Рисунок 2.14 – Приклад використання оператора Лапласа

Результат обробки графічного файлу за допомогою оператора Лапласа (рис. 2.15).

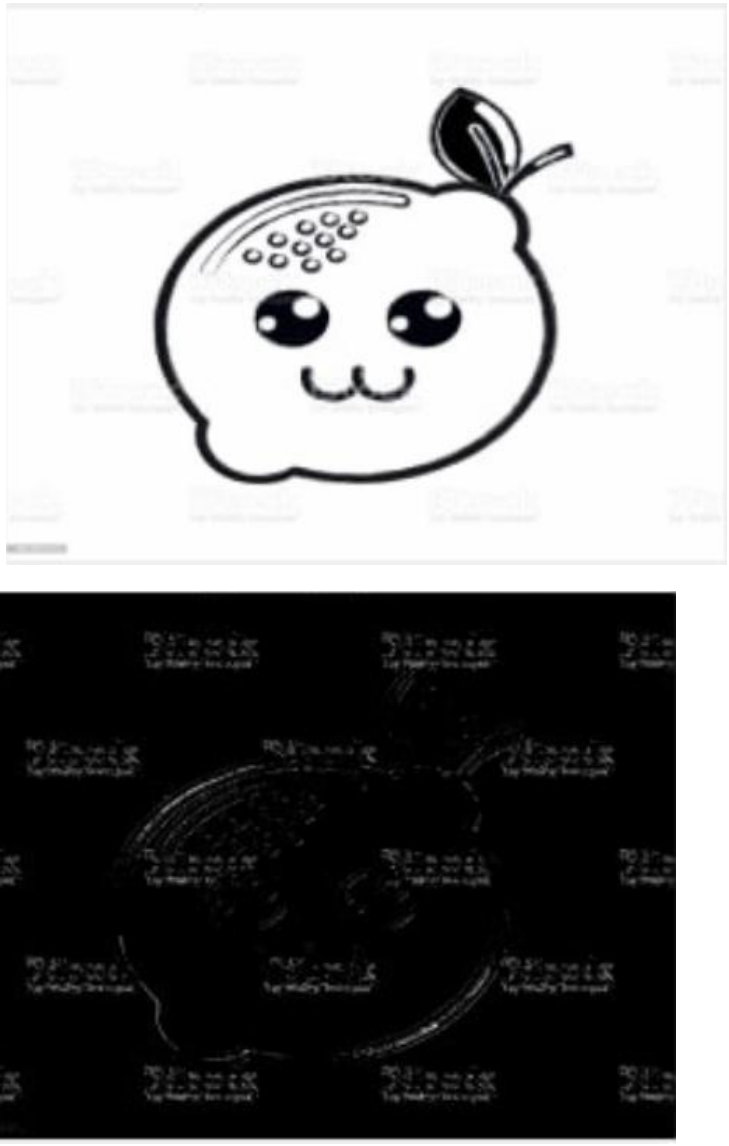


Рисунок 2.15 – Приклад роботи оператора Лапласа

Переваги:

1) займає мало ОП та має мало шагів;
2) легкий у використанні.

Недоліки:

1) недуже чітко обробляє малюнки та знаходить межі зображень.

## 2.2 Застосування алгоритмів.

## Використання алгоритмів для обробки зображення

Алгоритми, що були розроблені для взаємодії з графічними об’єктами використовуються не тільки в програмах. Також існують бібліотеки, котрі допомагають для згладжування, знаходження границь об’єктів та додаванню контраста або темних відтінків. В даному блоці представлено перелік цих бібліотек.

scikit-image — бібліотека Python з відкритим кодом, котра використовує алгоритм Собеля для обробки зображення. Ця бібліотека може використовувати сегментацію для геометричних зображень, змінювати кольори, аналізувати графічний об’єкт, фільтрація предметів на картинці. scikit-image взаємодіє з числовими бібліотеками такими як NumPy та SciPy. Приклад цієї бібліотеки (рис 2.16, 2.17, 2.18) демонструє фільтрацію зображення.

Переваги:

- легка для застосування;
- має високоякісний і рецензований код;
- хороша документація з прикладами.

Недоліки:

- займає багато пам’яті;
- необхідно використовувати інші бібліотеки, тобто не самостійна.

```
import matplotlib.pyplot as plt

from skimage import data, filters

image = data.coins ()

edges = filters.sobel (image)

plt.imshow (edges, cmap = 'gray')
```

Рисунок 2.16 — Використання бібліотеки scikit-image

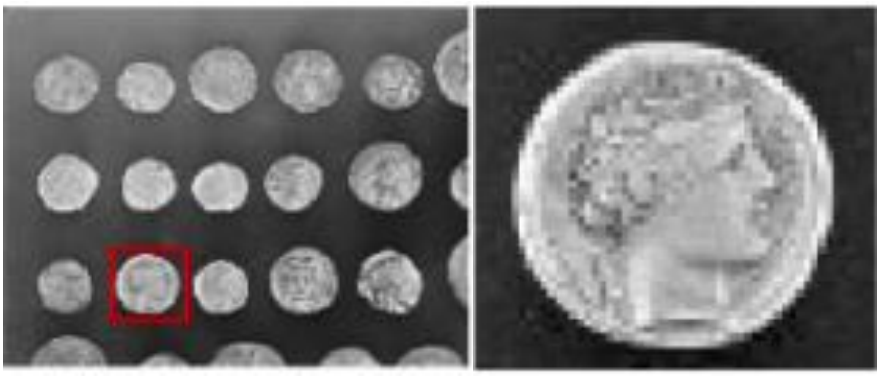


Рисунок 2.17 – Розпізнання та збільшення масштабу

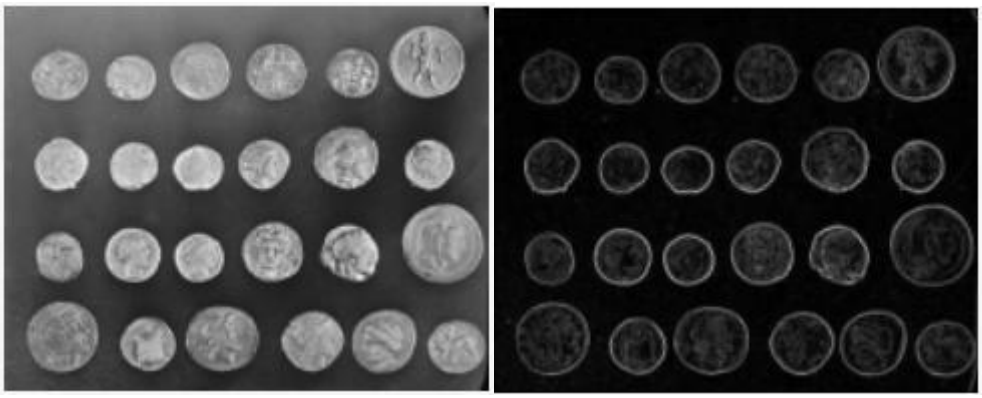


Рисунок 2.18 – Фільтрація малюнка

NumPy- це основа для багатьох інших бібліотек та має підтримку масивів. Це одна з основних Python-бібліотек з підтримкою масивів.

Зображення – це масив пікселів. Отже, за допомогою бібліотеки можна зробити такі операції з пікселями: маскування, зріз пікселів та індексувати. Коли, наприклад, всі пікселі проіндексовані можна змінювати зображення. Це вже робить інші бібліотеки.

Переваги:

- хороша документація с пікселями;
- основа для деяких інших бібліотек.

Недоліки:

- необхідно використовувати додаткові бібліотеки.

Приклад роботи бібліотеки NumPy (рис 2.19) – маскування зображення. Результати отримано на малюнку 2.20.

```
import numpy as np
from skimage import data
import matplotlib.pyplot as plt
% Matplotlib inline
image = data.camera ()
type (image)
numpy.ndarray # Зображення - це масив NumPy
mask = image <87
image [mask] = 255
plt.imshow (image, cmap = 'gray')
```

Рисунок 2.19 — Приклад роботи бібліотеки NumPy

Рисунок 2.20 – Накладання маски на предмет

SciPy – це модуль в Python подібний до NumPy. Має подібний функціонал. Але має можливість працювати з багатовимірними масивами. Працює на основі алгоритму Лапласа

Переваги:

- обробляє n-мірні масиви;

Приклад використан

ня розглядається на (рис 2.22) та зображує розмиття малюнку(рис 2.21).

```
from scipy import misc, ndimage

face = misc.face ()

blurred_face = ndimage.gaussian_filter (face, sigma = 3)

very_blurred = ndimage.gaussian_filter (face, sigma = 5)
```

Рисунок 2.21 — Приклад роботи бібліотеки SciPy

Рисунок 2.22 — Результат роботи програми

PIL (Python Imaging Library) – це бібліотека Python, яка містить в собі базовий функціонал для обробки зображень, такі як: фільтри, перетворення кольору та додавання контрасту. Використовується алгоритм Канні.

Переваги:

- безкоштовна бібліотека;
- підтримка різних форматів зображення;
- відкритий код;
- хороша документація з прикладами;
- легко встановити.

Недоліки:

- бібліотека більше не підтримується та не будуть виходити її оновлення.

Наведемо приклад для поліпшення зображення через ImageFilter в Pillow (рис 2.23). Результат роботи задання контрасту (рис 2.24):

```
from PIL import Image, ImageFilter
# Читання зображення
im = Image.open ( 'image.jpg')
# Відображення зображення
im.show ()
from PIL import ImageEnhance
enh = ImageEnhance.Contrast (im)
enh.enhance (1.8) .show ( "30% more contrast")
```

Рисунок 2.23 - Приклад роботи бібліотеки PIL

Рисунок 2.24 – Результат роботи

OpenCV (Open Source Computer Vision Library) – це Python бібліотека, що використовується у додатках для Computer Vision. Так як внутрішня структура бібліотеки розроблена на C та C++, то він працює швидко. А самі запити, що реалізує користувач виконується через знайомий Python, що додає легкість в користуванні[24].

Переваги:

- найпопулярніша;
- швидка;
- має відкритий код;

- хороша бібліотека для комп’ютерного зору;
- хороша бібліотека.

Приклад використання бібліотеки можна розглянути на поєднанні Яблука та Апельсину (рис 2.25).

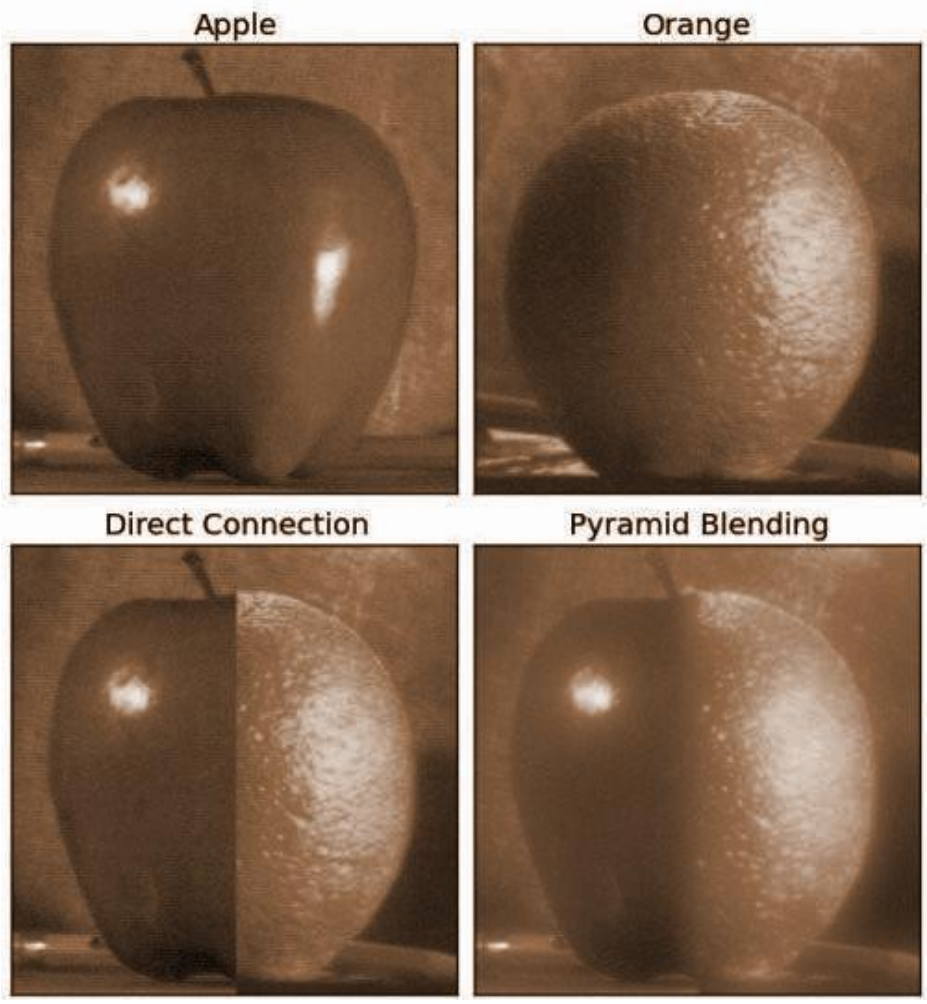



Рисунок 2.25 – Поєднання рисунків

SimpleCV – як і OpenCV використовується у додатках, що працюють з комп’ютерним зором. Працює через запозичення функціоналу у інших бібліотек комп’ютерного зору. Бібліотека працює так, що не потрібно аналізувати глибину кольору або колірний простір графічного файлу це дає можливість ще легшої обробки зображення.

Розглянемо приклад роботи на виділення контурів зображення (рис 2.26).

Рисунок 2.26 — Приклад роботи бібліотеки SimpleCV

Переваги:

- легкий у використанні;
- працює з відеофайлами;
- зрозуміла документація з прикладами.

Недоліки:

- повільніший за OpenCV.

Mahotas - це бібліотека для комп’ютерного зору та обробки зображень. Для обробки інформації, що є в картинці, ця бібліотека має такі функції: фільтри для корекції зображення, морфологічні операції, виявлення особливих точок та локальні дескриптори. Базується на C++, що надає переваги в швидкості. Також має мінімалістичний код та мало залежностей.

Переваги:

- швидка;
- хороша документація.

Розглядається результат роботи бібліотеки для пошуку. (Пошук героя з мультфільма “Де Воллі,”)

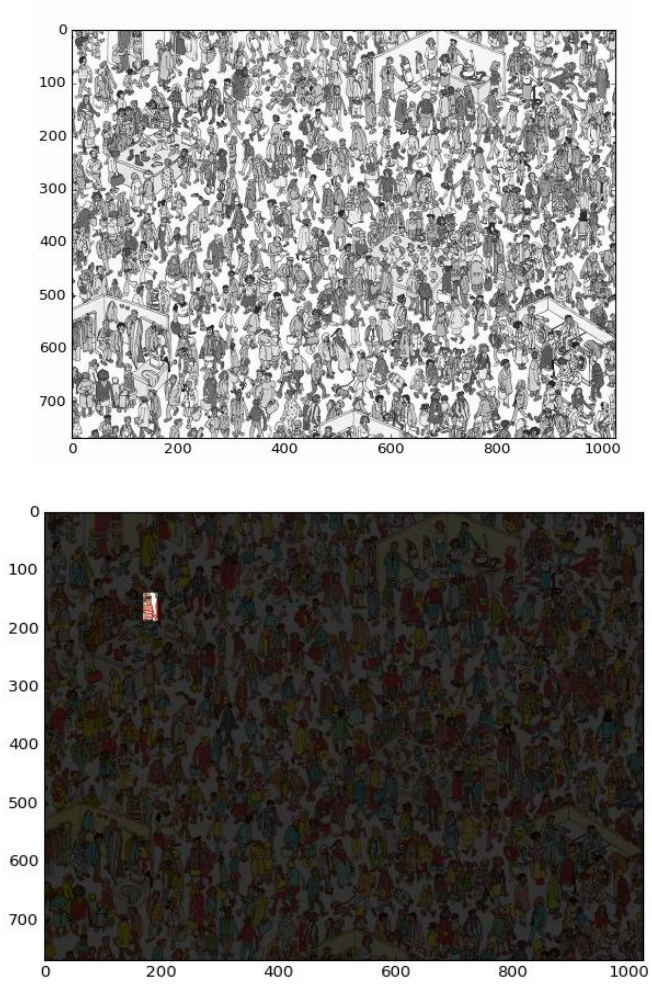



Рисунок 2.27 — Результат роботи Mahotas

Pgmagick - це обгортка для система GraphicsMagick. Бібліотека багатофункціональна. GraphicsMagick підтримує обробку, читання та запис з різними форматами графічних файлів. Таких як: DPX, GIF, JPEG, JPEG-2000, PNG, PDF, PNM і TIFF. Дана бібліотека використовує алгоритми Канні та Собеля та фільтрацію Гауса[25].

Переваги:

- працює з багатьма форматами графічних файлів;
- має багато функцій.

Недоліки:

- не дуже швидко працює;
- потрібно багато пам’яті.

Приклади використання коду для загострення і фільтрації (рис 2.28) та демонстрація роботи (рис 2.29).

```
from pgmagick.api import Image

img = Image('ouroku.jpg')
img.sharpen(1)
img.write('ouroku_sharpen1.jpg')
```

Рисунок 2.28 — Приклад коду на Python


Рисунок 2.29 — Демонстрація роботи коду для загострення

Приклади використання для визначення та виділення меж об’єктів на зображенні. (рис. 2.30) та (рис. 2.31).

```
from pgmagick.api import Image

img = Image('ouroku.jpg')
img.edge(2)
img.write('ouroku_edge.jpg')
```

Рисунок 2.30 — Приклад коду на Python


Рисунок 2.31 — Визначення меж об’єктів

Приклади використання бібліотеки для намалювання тексту (рис 2.32) та (рис 2.33) як результат.

```
from pgmagick.api import Image
img = Image((300, 200))
img.font("/usr/share/fonts/truetype/ttf-japanese-gothic.ttf")
img.annotate('Hello World')
img.annotate('ようこそpgmagickへ!!')
img.write('japanese-text.png')
```

Рисунок 2.32 — Приклад коду на Python


Рисунок 2.33— Результат роботи

PyCairo — являє собою набір прив'язок Python-коду для графічної бібліотеки Cairo. Cairo - це 2D-бібліотека для відтворення векторної графіки, так як векторна графіка не втрачає якість, чіткість при зміні розмірів чи трансформації малюнка. Cairo-команди працюють за допомогою прив’язки до PyCairo. Використовується фільтрація Гауса.

Переваги:

- швидко працює.

Недоліки:

- працює тільки з векторною графікою.

Приклад роботи бібліотеки розглядається через заливку прямокутників у різні кольори код на мові Python (рис 2.34) та демонстрація роботи (рис 2.35).

```
import gi
gi.require_version('Gtk', '3.0')
from gi.repository import Gtk
import cairo


def on_draw(self, wid, cr):

    cr.set_source_rgb(0.2, 0.23, 0.9)
    cr.rectangle(10, 15, 90, 60)
    cr.fill()

    cr.set_source_rgb(0.9, 0.1, 0.1)
    cr.rectangle(130, 15, 90, 60)
    cr.fill()

    cr.set_source_rgb(0.4, 0.9, 0.4)
    cr.rectangle(250, 15, 90, 60)
    cr.fill()


cr.set_source_rgb(0.2, 0.23, 0.9)
cr.rectangle(10, 15, 90, 60)
cr.fill()
```

Рисунок 2.34 — Приклад коду на Python

Рисунок 2.35 — Демонстрація роботи

SimpleITK або Insight Segmentation and Registration Toolkit - це крос-платформна система з відкритим кодом, що має в собі розширений набір інструментів для аналізу зображень. А ось SimpleITK — це спрощений вигляд ITK, який використовує Insight Segmentation and Registration Toolkit та спрощує працю з цим інструментом. SimpleITK полегшує роботу з бібліотекою при швидкій взаємодії компонентів між собою та типізацією функцій, що використовує ITK. За допомогою навчання і інтерпретованим мовам. Тобто можна сказати, що SimpleITK — це набір інструментів для аналізу зображень з великою кількістю компонентів, що підтримують загальну фільтрацію, сегментацію і реєстрацію зображень. Сам SimpleITK написаний на C ++, але доступний для багатьох мов програмування, включаючи Python.

Переваги:

- швидко працює;
- легкий у використанні;
- багатофункціональний;
- можливість додавання C++.

Недоліки:

- витрачає багато пам’яті;
- не вигідно використовувати для буденних справ.

Демонстрація роботи SimpleITK для аналізу картинки та руху її компонентів (рис. 2.36).

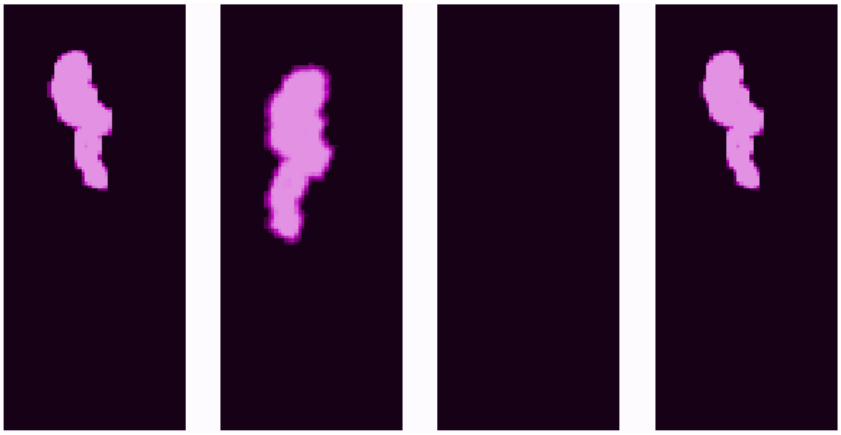


Рисунок 2.36 — Демонстрація роботи SimpleITK

## Висновки до другого розділу

В даному розділі розглянуто різні методи та алгоритми для обробки зображень та знаходження меж об’єктів. Були знайдені їх недоліки та переваги. Порівняння між собою та виявлення кращих.

Опрацювавши розділ 2, можна зробити наступні висновки: оператор Собеля працює краще, а от алгоритм Канні швидше. Також можливо простежити схожість усіх алгоритмів між собою. Тобто частіше за все алгоритми визначення меж об’єктів працюють по однаковим крокам. І відрізняються лише в методах вирішення цих кроків. Наприклад, знаходження границь можна робити через фільтр Гауса або через знаходження похідної.

Також розділ розкриває необхідність у модифікації алгоритма Канні. Тобто описом даного алгоритма відкриваються його недоліки та необхідність його виправлення.

Ще в розділі вказанні бібліотеки в яких використовуються алгоритми для обробки зображення, тобто продемонстрована необхідність для використання у даній магістерській роботі.

# 3 СПОСІБ МОДИФІКАЦІЇ АЛГОРИТМУ КАННІ

## 3.1 Вибір середовища

Для даного проєкту було обрано C# для наглядної демонстрації роботи алгоритму.

C# (рис 3.1) - це сучасна мова програмування, яку можна використовувати для різних задач. Найпоширеніша платформа використання - Windows .NET. C# - це об’єктно-орієнтована мова програмування (ООП)[15]. Так як ця мова має статичний вигляд, про помилки, що є в програмі, людина дізнається до компіляції. Це дозволяє не витрачати час на відловлювання помилок.  C# вважається мовою високого рівня, тому її легко читати та писати на ній.

Рисунок 3.1 - Емблема мови C#

C# використовується в таких сферах: автоматизація важких процесів, написання додатків для .NET платформ, написання ігор на движку Unity, робота з базами даних, бізнес аналіз, написання корпоративних програм.

### Переваги та недоліки C#

Переваги C#:

- наявність EntityFramework: підтримується Microsoft, легша можливість міграцій модулів;

- нещодавно з’явилась підтримка Linux та MacOS за допомогою .NET Core;
- активний розвиток Xamarin;
- розвиток в напрямку Internet of things, IoT (концепція мережі передачі між фізичними речами);
- він має величезну бібліотеку, яка може забезпечити набагато більшу функціональність високого рівня, ніж інші мови, такі як Java та C++;
- нативні блоки коду та керовані блоки коду підтримуються в C#.
- має гарну інтеграцію додатків Windows.
- легкий для початкового використання.

Недоліки C#:

- компіляція залежить від часу роботи операцій;
- VisualStudio - платформа за допомогою якої здійснюється компіляція повільна;
- технологія Windows Mobile має свої недостатки.

Різниця між Java та Python та C#

Проведено аналіз популярних мов програмування, щоб визначити на якій краще за все демонструвати роботу алгоритму.

Java (рис 3.2) - високорівнева, об’єктно-орієнтована мова програмування. Платформа Java - обчислювальна платформа для розробки додатків. Java використовується в таких напрямках: корпоративне програмне забезпечення, обчислювальні програми, аналіз даних, програмування апаратних пристроїв, забезпечення сервісні технології(Apache, JBoss, GlassFish), написання додатків для Andoid[27].


Рисунок 3.2 - Емблема Java

Переваги та недоліки Java

Переваги Java:

- різні варіанти як збирати проєкт;
- багато інформації та форумів в інтернеті;
- наявність OpenJDK;
- стабільність;
- можливість використовувати в фінансових системах;
- використовує IDE IntelliJ IDEA.

Недоліки Java:

- відсутність хорошої документації які бібліотеки для чого необхідні;
- погана API для роботи з датами та часом;
- Maven дуже повільно працює;
- повільно розвивається API;
- відсутність функцій вищого порядку та альтернативи LINQ;

Переваги C# за Java.

C# - це покращена Java, через наявність auto-property, використовується легка модель асинхронного програмування, присутній LINQ, що покращує реалізацію провайдерів, таких як LINQ to SQL, LINQ to XML тощо.

Для проєктів на Java відсутнє використання нестандартних та різних рішень. Все однозначне та типове. Java поглинає більше ресурсів комп’ютера та ОП.

Python (рис 3.3) - це високорівнева мова програмування загального призначення. Python можна використовувати в таких сферах: розробка веб- та мобільних додатків, обробка великих даних, математичні обчислення, написання системних скриптів (автоматизація роботи) та для інженерії програмного забезпечення.


Рисунок 3.3 - Емблема Python

Переваги Python:

- можна використати об'єктно-орієнтоване програмування, функціональне та структурне програмування;
- легко кодувати, читати, підтримувати та портувати;
- можливість використовувати на Windows, Mac або Unix;
- велика кількість бібліотек;
- пддтримує автоматичний збір сміття;
- взаємодія з різними мовами програмування (можна імпортувати та застосовувати інші мови програмування, коли пишеш код на Python);
- підходить для мережевих програм, які використовують кілька протоколів.

Недоліки Python:

- повільно працює;
- Python вважають хорошим тільки через бібліотеки. Без бібліотек не має багато функцій, котрі необхідні;

- Global Interpreter Lock не дає можливість використовувати багатопоточність;
- погано працює багатопоточність в програмах;
- поганий для розробки мобільних додатків;
- необхідно багато пам’яті для запуску.

Переваги C# за Python.

C# має більш організовану структуру. Це означає, що немає невідповідностей у синтаксисі та правилах форматування. Тобто набагато легше працювати та побудувати додатки. C# краще видає графічні порівняння. Коли користувач буде визначатися, котрий алгоритм використовувати, C# буде у вигідніших умовах, тому що використовує мало пам’яті та скоріше працює.

Отже, для відображення роботи модифікованого алгоритму Канні та порівняння його з вже існуючими, автор проєкту вирішила використовувати C#.

## 3.2 Алгоритми визначення границь. Початкові дані

Для визначення меж об'єкта методом Канні ми повинні виконати кілька операцій, які виконуються послідовно в кожному з алгоритмів, що використовуються зараз, це:

1) приведення зображення до відтінків сірого;
2) згладжування;
3) придушення не-максимумів;
4) подвійна порогова фільтрація;
5) трасування областей неоднозначності.

У кожній з операцій можливі алгоритми обробки. Розглянемо всі етапи цих операцій та виберемо найоптимальніший варіант обробки.

У даній реалізації алгоритму більшість цих кроків буде замінено, нехай і неявно на модифікований алгоритм, наприклад – придушення не-максимумів замінюється на знаходження похідної різкості, згладжування, яка являється коефіцієнтом порівняння потужності. Подвійна порогова фільтрація, що здійснюється за допомогою операції Собеля по вертикалі та горизонталі буде виконана шляхом одного порівняння різкості пікселя із сусідніми.

Надалі вважатиметься, що з обробки вибраного цифрового зображення розміром *w*h* пікселів, що з математичного погляду можна вважати двомірною матрицею *I*. Елемент цієї матриці *I*(*i, j*) містить ціле значення яскравості $f \in (0,255)$.

## 3.3 Переведення зображення у відтінки сірого

Для виділення контуру необхідно вихідне цифрове зображення *I*[*i,j*] перетворити на монохромне *G*[*i, j*].

При наведенні зображення до відтінків сірого можливе використання одного з алгоритмів:

- множення кожної із складових кольору (RGB) на певні коефіцієнти:

$$g = 0.299 * r + 0.587 * g + 0.114 * b \quad (24)$$

- обчислення середнього арифметичного із трьох складових кольору:

$$g = \frac{r + g + b}{3} \quad (25)$$

Алгоритмічне (з прикладу мови C#) уявлення цих формул:

- використання коефіцієнтів:

```
byte gray = (byte) (0.299 * color.R + 0.587 * color.G + 0.114 * color.B);
```

- середнє арифметичне:

byte gray = (byte) ((color.R + color.G + color.B) / 3);

Наведемо результати перетворення для двох різних зображень (рис 3.4) та (рис 3.5).

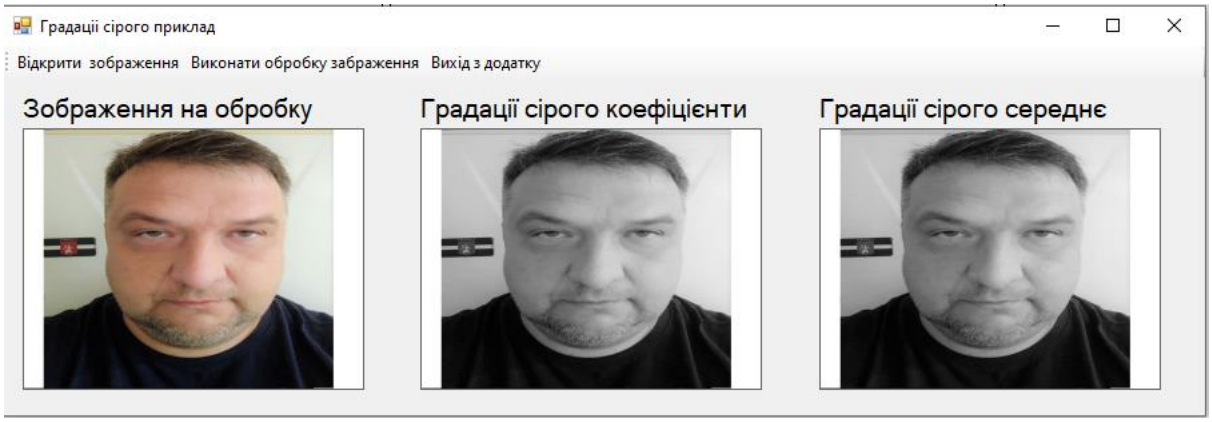



Рисунок 3.4 - Фотографія людини у градаціях сірого

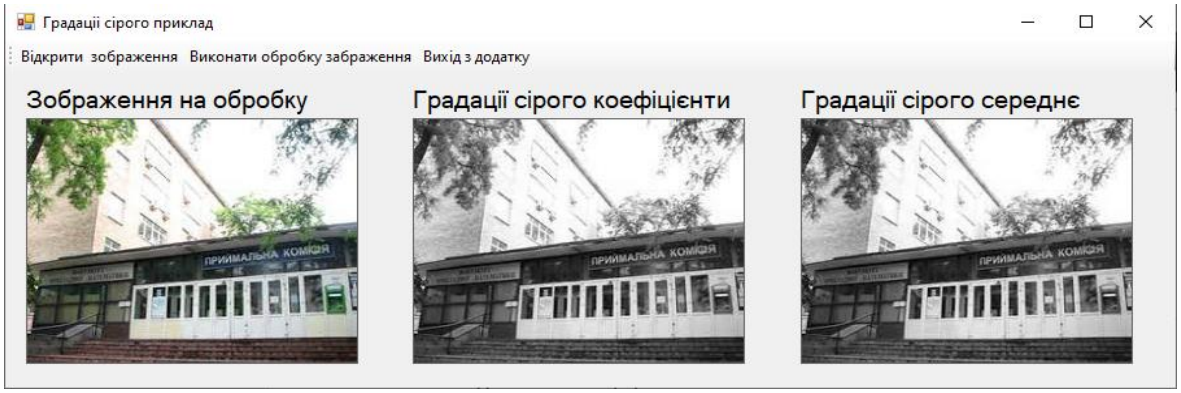



Рисунок 3.5 - Панорама у градаціях сірого

Як видно, обидва результати обробки приблизно однакові, також швидкість обробки на цьому етапі не відрізняється. Надалі ми будемо використовувати результати градації за середнім арифметичним, але при обчисленні швидкості обробки включимо обидва алгоритми для більш детального дослідження.

## 3.4 Виділення контуру методом обробки відтінків сусідніх пікселів

Ідея запропонованого методу полягає у розмитті «непотрібних» відтінків при переході від одного пікселя до іншого. Перехід від одного «потрібного» пікселя до іншого вважатиметься лінією контуру, решта – фон.

Результат обчислень зберігатимемо в бінарній матриці $B_{i,j}$, де значення 1 відповідає контуру, а значення 0 - фону.

Виділення контуру виконується за формулою:

$$B_{i,j} = \begin{cases} 1, & t_{i,j} \le G_{i,j} + |c_{i,j}| \\ 0, & t_{i,j} > G_{i,j} + |c_{i,j}| \end{cases} \quad (26)$$

У цій формулі множина $c_{i,j}$ складається з кольорів пікселя $(i,j)$ та кольорів пікселів його сусідів,

$$\mathrm{c}_{i,j} = \{G_{i+k,j+m}, -1 \le k \le 1, -1 \le m \le 1\} \quad (27)$$

а $t_{i,j}$- це середнє арифметичне елементів цієї множини.

$$t_{i,j} = \frac{1}{9}\sum_{-1\le m\le 1}^{-1\le k\le 1} G_{i+k,j+m} \quad (28)$$

В результаті обчислень за формулою (3) ми отримаємо контури, представлені на малюнку 3.3. Як і в детекторі кордонів Канні, ми згладжуватимемо зображення, але не за допомогою фільтра Гауса, а шляхом порівняння потужності множини $|c_{i,j}|$ з певним коефіцієнтом. Якщо потужність більше чотирьох, встановлюємо його рівною 4, що робиться для підвищення ймовірності виключення з лінії контуру.

Алгоритм мовою C# виглядає так (рис 3.7)

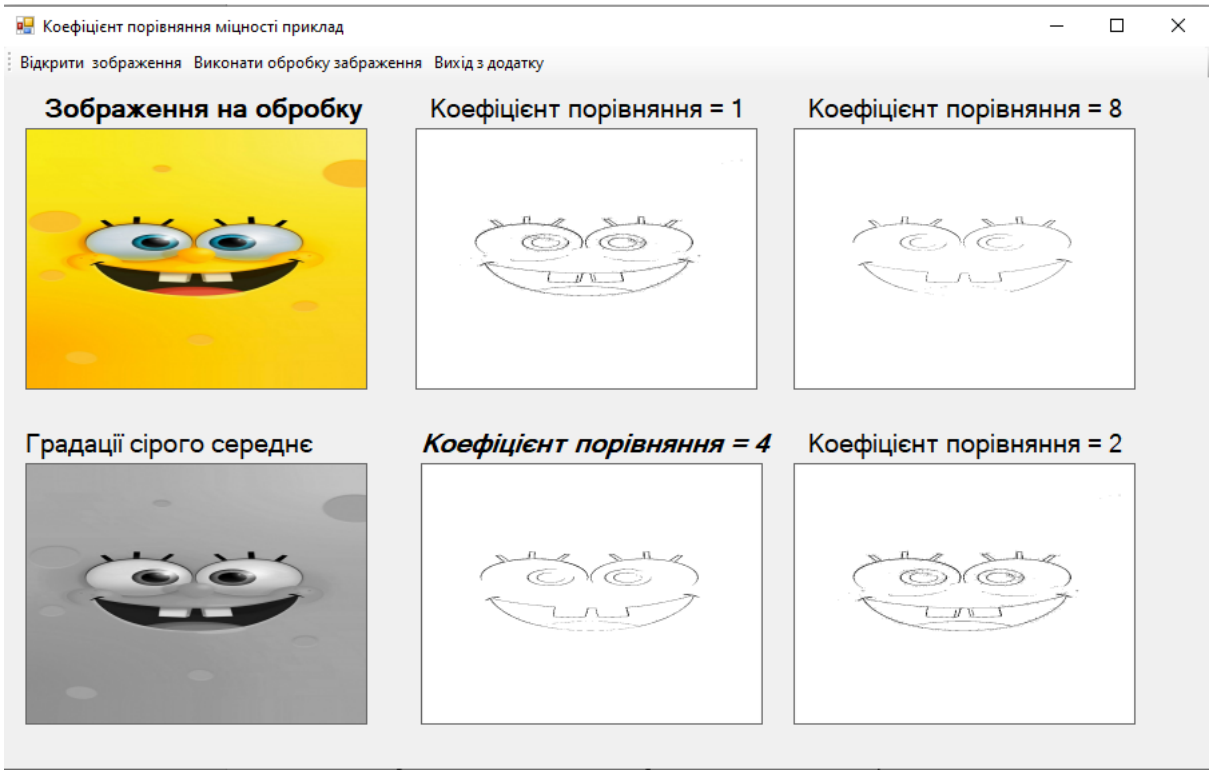



Рисунок 3.6 - Застосування різних коефіцієнтів для обробки простого зображення

```
for (int i = 1; i < Width - 1; i++)
        {
           for (int j = 1; j < Height - 1; j++)
           {
              s[0] = MatrixG[i - 1, j - 1];
              s[1] = MatrixG[i - 1, j];
              s[2] = MatrixG[i - 1, j + 1];
              s[3] = MatrixG[i, j - 1];
              s[4] = MatrixG[i, j];
              s[5] = MatrixG[i, j + 1];
              s[6] = MatrixG[i + 1, j - 1];
              s[7] = MatrixG[i + 1, j];
              s[8] = MatrixG[i + 1, j + 1];
       temp = (s[0] + s[1] + s[2] + s[3] + s[4] + s[5] + s[6] + s[7] +
s[8])/koefAverage;


 MatrixB[i, j] = temp <= (MatrixG[i, j] + Cardinality(s)) ? 255 : 0;
  }
 }
```

Рисунок 3.7 — Алгоритм мовою C#

Важливо відзначити, що накладення власних умов на потужність множини може призвести до різних результатів. Цю обставину можна застосувати, коли потрібно отримати більш специфічне рішення завдання виділення контурів. Аналіз найбільш відповідного коефіцієнта добре виконаний у роботі, що представлена у [22].

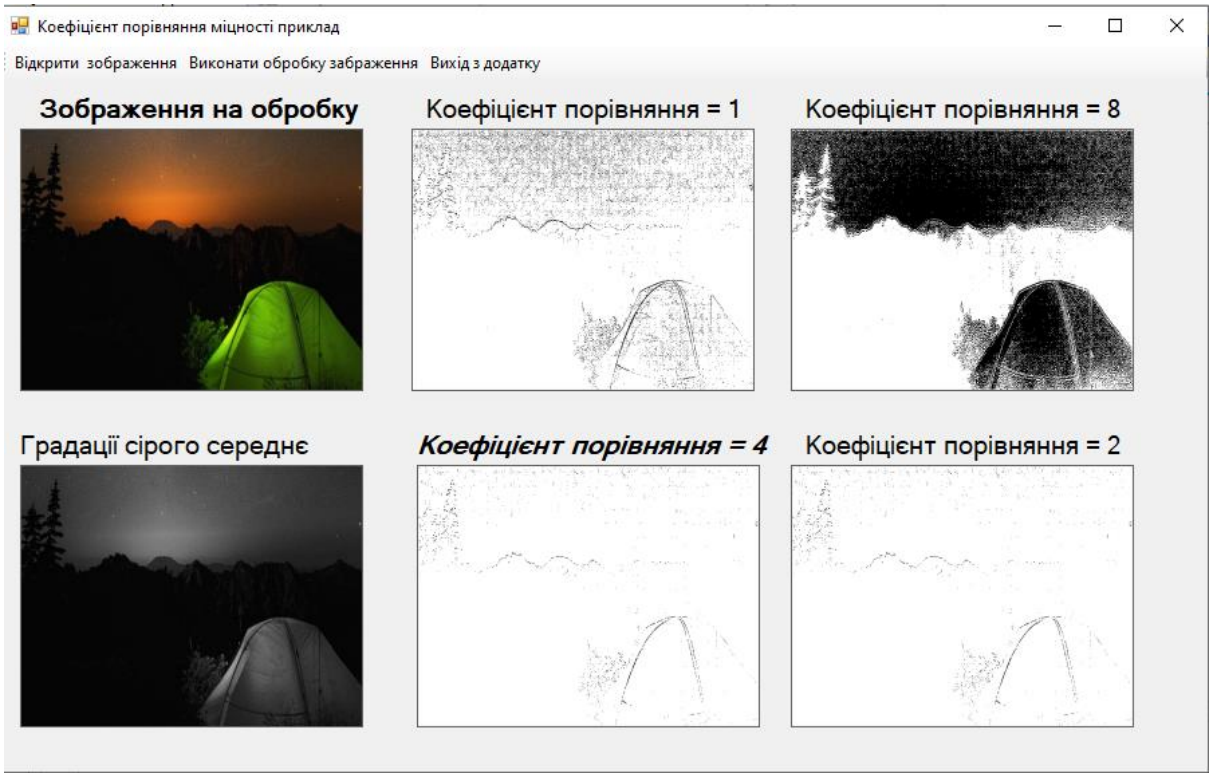



Рисунок 3.8 — Застосування різних коефіцієнтів для обробки панорамного зображення

Як видно з малюнка (рис 3.5), найбільш оптимальним значенням коефіцієнта порівняння потужності є 4. Цей показник означає, якщо піксель, що обробляється, відрізняється більш, ніж від половини сусідніх пікселів (з будь-якого боку), то вага цієї різниці не має значення. У таблиці 3.1 наведено кілька варіантів підрахунку даного коефіцієнта для різних значень кольору навколишніх пікселів:

Таблиця 3.1- Приклади підрахунку коефіцієнта приведення потужностей

| Матриця навколишня піксель (значення кольору) | Врахування значення при підрахунку | Коефіцієнт (кількість відмінних пікселів) |
|---|---|---|
| 1 | 2 | 3 |

| | | |
|---|---|---|
| 10 12 12<br>12 12 12<br>5 7 8 | 1 0 0<br>0 0 0<br>1 1 1 | 4 (4) |
| 12 12 12<br>12 12 12<br>12 7 8 | 0 0 0<br>0 0 0<br>0 1 1 | 2(2) |
| Продовження таблиці 3.1 | | |
| 1 | 2 | 3 |
| 10 10 10<br>12 12 12<br>5 12 8 | 1 1 1<br>0 0 0<br>1 0 1 | 4(5) |
| 10 1 11<br>12 12 12<br>5 7 8 | 1 1 1<br>0 0 0<br>1 1 1 | 4(6) |

Для покращення якості розпізнавання, розглянемо можливість попередньої обробки зображення за допомогою деякого предиктора, який скоригує різкість вихідного зображення. Декілька різних способів передобробки було запропоновано автором роботи [23]. У цьому проєкті буде запропоновано новий спосіб, що базується на приведенні яскравості пікселя до деякого значення.

## 3.5 Застосування похідної різкості

Представлено результати обробки простого зображення (рис. 3.3), з нього видно, що в даному випадку коефіцієнт приведення потужностей сусідніх пікселів не має суттєвого значення.

Ця обставина, у тому числі, пов'язана з невеликою кількістю кольорів та їх відтінків у матриці відтінків сірого.

Для підвищення якості деталізації зображення при виділенні контурів необхідно виконати попередню обробку зображення, яка б усереднила яскравість усіх пікселів до певної константи, причому інформативність вихідного зображення не повинна зменшитися.

Людське око може визначити близько 30 відтінків сірого залежно від освітлення [24]. Ґрунтуючись на цьому, можна усереднити зони зображення до конкретного сірого відтінку, виходячи з поточної яскравості пікселя. Однак бітова глибина кольору (модель RGB) дорівнює 24 бітам, яка не дозволяє коректно згрупувати 256 відтінків в 30, тому розіб'ємо колірний спектр монохромного зображення на 24 групи. Таким чином, кожна група складатиметься з 256/24=10,67≈11 відтінків. Потім в оброблюваному монохромному зображенні проводиться заміна відтінків, що потрапляють у групу, на початковий відтінок цієї групи (таблиця 3.2).

Таблиця 3.2 - Заміни відтінків

| **Початкові відтінки** | **Нові відтінки** | **Початкові відтінки** | **Нові відтінки** | **Початкові відтінки** | **Нові відтінки** |
|---|---|---|---|---|---|
| 0..10 | 0 | 88..98 | 88 | 176..186 | 176 |
| 11..21 | 11 | 99..109 | 99 | 187..197 | 187 |
| 22..32 | 22 | 110..120 | 110 | 198..208 | 198 |
| 33..43 | 33 | 121..131 | 121 | 209..219 | 209 |
| 44..54 | 44 | 132..142 | 132 | 220..230 | 220 |
| 55..65 | 55 | 143..153 | 143 | 231..241 | 231 |
| 66..76 | 66 | 154..164 | 154 | 242..252 | 242 |
| 77..87 | 77 | 165..175 | 165 | 253..255 | 253 |

Програмна формула для обчислення нового відтінку має вигляд:

```
s[i] = (int)Math.Floor((decimal)s[i]/(decimal)koefGradation) * koefGradation;
```

Як бачимо, ми можемо змінювати кількість відтінків у групі для того, щоб знайти найбільш оптимальне значення коефіцієнта похідної.

Слід зазначити, що при обробці «простих» зображень цей коефіцієнт не відіграє великого значення (рис. 3.9), тоді як для обробки панорамних зображень цей коефіцієнт має більш видиме значення (рис.3.10) і найбільш оптимальний результат показує коефіцієнт 11.

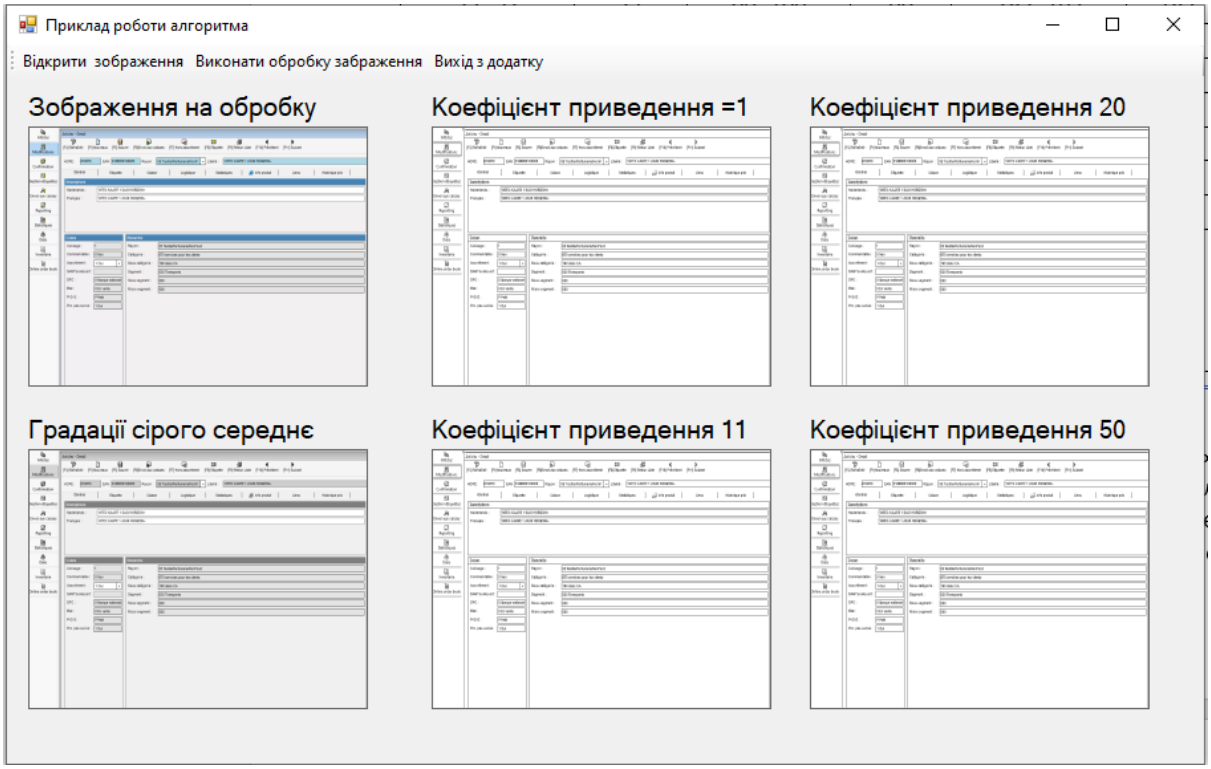



Рисунок 3.9 — Застосування різних коефіцієнтів приведення різкості для обробки простого зображення

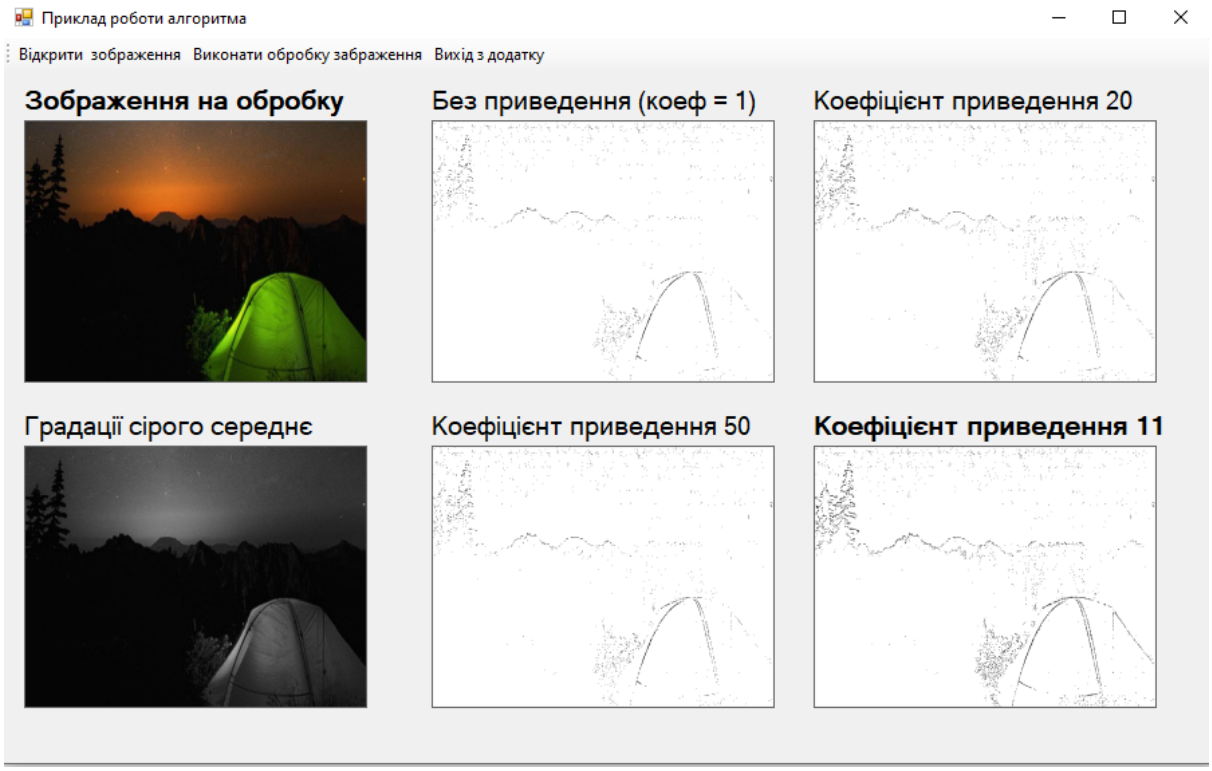



Рисунок 3.10 — Застосування різних коефіцієнтів приведення різкості під час обробки панорамного зображення

## Висновки до третього розділу

У розділі детально розглянуті всі кроки для визначення меж зображень за допомогою алгоритму Канні. Також наведена модифікація цих кроків. Показані програмні рішення та алгоритм модифікації автора проєкту.

Визначено необхідні завдання, які ставить перед собою модифікований алгорим.

Продемонстровано в програмному коді покрокова робота модифікованого алгоритму.

# 4 ПОРІВНЯННЯ ЗАПРОПОНОВАНОГО АЛГОРИТМУ

## 4.1 Порівняння операцій нового алгоритму з алгоритмом Кані

У першому розділі докладно описано роботу алгоритму визначення меж методом Канні, показано формули реалізації кожного кроку в його реалізації. У таблиці 4.1 розглянуто ці кроки та як ці етапи обробки реалізуються у новому алгоритмі.

Таблиця 4.1 – Етапи обробки зображення

| Етап | Визначення кордонів методом Кані | Новий алгоритм |
|---|---|---|
| 1 | 2 | 3 |
| Перетворення на відтінки сірого | Формули: $RGB2Yuv := \begin{pmatrix} 0.299 \\ 0.587 \\ 0.114 \end{pmatrix}$ $rRGBMatrix := submatrix\left(RGBMatrix, 0, rows(RGBMatrix)-1, 0, \frac{cols(RGBMatrix)}{3}-1\right)$ $gRGBMatrix := submatrix\left(RGBMatrix, 0, rows(RGBMatrix)-1, \frac{cols(RGBMatrix)}{3}, \frac{cols(RGBMatrix)\cdot 2}{3}-1\right)$ $bRGBMatrix := submatrix\left(RGBMatrix, 0, rows(RGBMatrix)-1, \frac{cols(RGBMatrix)\cdot 2}{3}, cols(RGBMatrix)-1\right)$ | $g = \frac{r+g+b}{3}$ |
| Згладжування | $G\_gen(x,y,\sigma) := \frac{1}{2\pi\cdot\sigma^2}\cdot e^{\frac{-(x^2+y^2)}{2\sigma^2}}$ $CrtGauFiltKern(size,\sigma) := \left\| \begin{array}{l} Shift \leftarrow floor\left(\frac{size}{2}\right) \\ \text{for } y \in 0..size-1 \\ \quad \text{for } x \in 0..size-1 \\ \quad\quad M_{y,x} \leftarrow G\_gen(x-Shift, y- \\ \text{return } M \end{array} \right.$ | Коефіцієнт згладжування дорівнює 4 |

Продовження таблиці 4.1

| 1 | 2 | 3 |
|---|---|---|
| Пошук градієнтів | $MGx := \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix}$ $MGy := \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$<br>SobelOperator(Matrix) := for iY ∈ 1.. rows(Matrix) − 2<br>for iX ∈ 1.. cols(Matrix) − 2<br>A ← submatrix(Matrix,iY − 1,iY + 1,iX<br>$GX \leftarrow \sum_{y=0}^{2} \sum_{x=0}^{2} \left(A_{y,x} \cdot MGx_{y,x}\right)$<br>$GY \leftarrow \sum_{y=0}^{2} \sum_{x=0}^{2} \left(A_{y,x} \cdot MGy_{y,x}\right)$<br>$G \leftarrow \sqrt{GX^2 + GY^2}$<br>$\theta \leftarrow \text{round}\left(\frac{\text{atan2}(GX,GY)}{\frac{\pi}{4}}\right) \cdot \frac{\pi}{4} - \frac{\pi}{2}$ if<br>θ ← ErrCode otherwise<br>$SobMtrx_{iY,iX} \leftarrow G$<br>$SobMtrx_{iY,iX+1+cols(Matrix)} \leftarrow \theta$<br>return SobMtrx | Порівняння потужності пікселя із сусідніми |
| Придушення не-максимумів | IsCorrectIndex(Matrix,x,y) := return 0 if x < 0 ∨ x > cols(Matrix) - 1 ∨ y < 0 ∨ y > rows(Matrix) - 1<br>return 1<br>Check(Matrix,x,y,v) := return 0 if IsCorrectIndex(Matrix,x,<br>return 1 if $Matrix_{y,x} \le v$<br>return 0<br>NonMaximumSuppression(Matrix) := ANGmatrix ← GetRightSubMatrix(Matrix)<br>Gmatrix ← GetLeftSubMatrix(Matrix)<br>for y ∈ 0.. rows(Gmatrix) - 1<br>for x ∈ 0.. cols(Gmatrix) - 1<br>continue if $ANGmatrix_{y,x} = ErrCode$<br>$dx \leftarrow sign\left(\cos\left(ANGmatrix_{y,x}\right)\right)$<br>$dy \leftarrow -sign\left(\sin\left(ANGmatrix_{y,x}\right)\right)$<br>$RMatrix_{y+dy,x+dx} \leftarrow 0$ if Check(Gmatrix,x + dx,y + dy<br>$RMatrix_{y-dy,x-dx} \leftarrow 0$ if Check(Gmatrix,x - dx,y - dy,<br>$RMatrix_{y,x} \leftarrow Gmatrix_{y,x}$<br>return RMatrix | Не використовується |

| | | |
|---|---|---|
| Подвійна порогова фільтрація | DoubleThresholding(Matrix,low_pr,high_pr) := down ← low_pr·255<br>up ← high_pr·255<br>for y ∈ 0..rows(Matrix) - 1<br>for x ∈ 0..cols(Matrix) - 1<br>RM$_{y,x}$ ← 255 if Matrix$_{y,}$<br>otherwise<br>RM$_{y,x}$ ← 0 if Matrix$_{y}$<br>RM$_{y,x}$ ← 127 otherwi<br>return RM | Застосування лімітів збільшення при аналізі потужності сусідніх пікселів |

Відображено результати обробки зображення методом Кані та новим алгоритмом (рис 4.1). Як видно, вони мало відрізняються і можуть бути використані у подальшій практичній роботі.

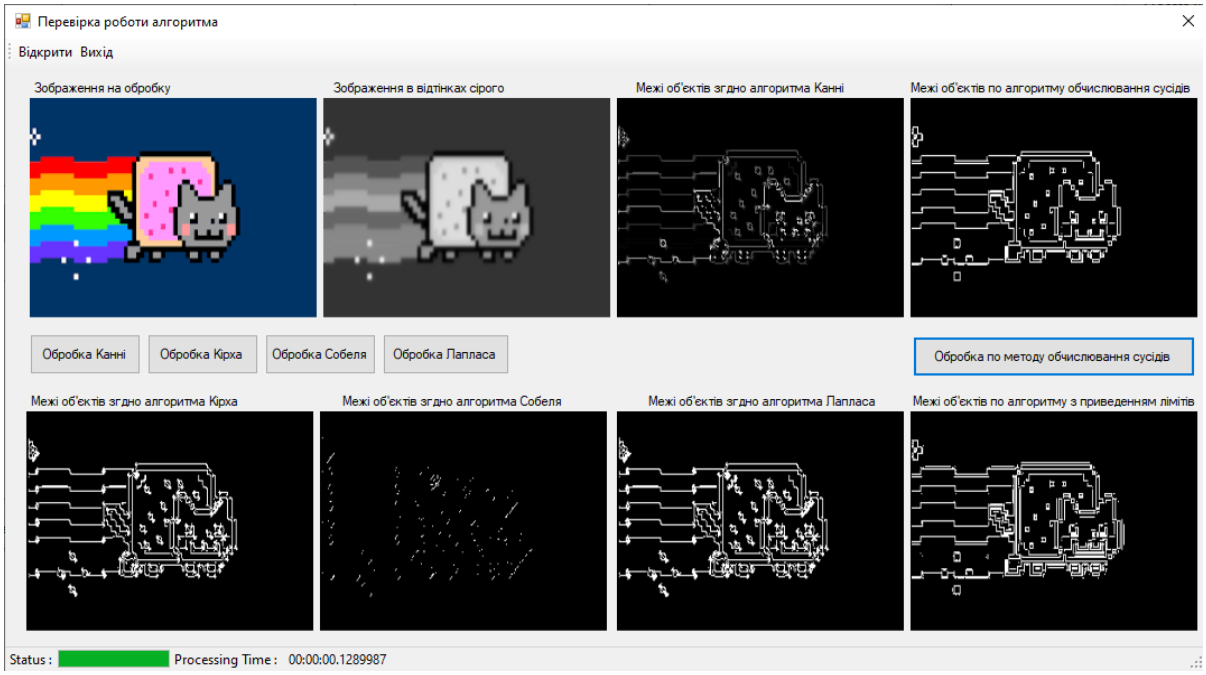

Рисунок 4.1 – Результати обробки зображення

## 4.2 Порівняння з іншими алгоритмами

Проведемо порівняння запропонованого підходу з іншими популярними алгоритмами виділення контуру, такими як оператори Канні та Кірша, а також Лапласіаном 5x5. Порівняння результатів виконання алгоритмів проведемо за такими оцінками:

- візуальна оцінка якості виділення контурів за допомогою нечітких множин;
- оцінка товщини контурів отриманих зображень;
- чисельна оцінка якості виділення меж на основі пікового відношення сигналу до шуму;
- оцінка тимчасових витрат за виконання алгоритмів.

Візуальна оцінка. Для тестування якості роботи алгоритму проведено порівняння з іншими алгоритмами різних зображеннях.



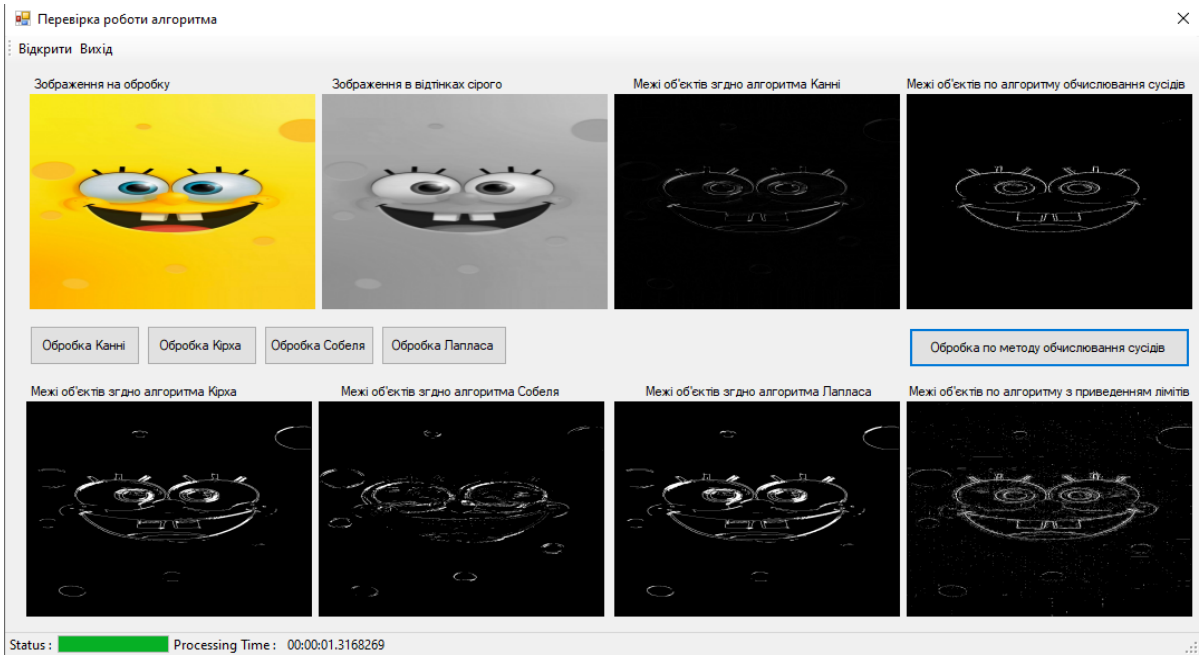



Рисунок 4.2 - Обробка простого зображення

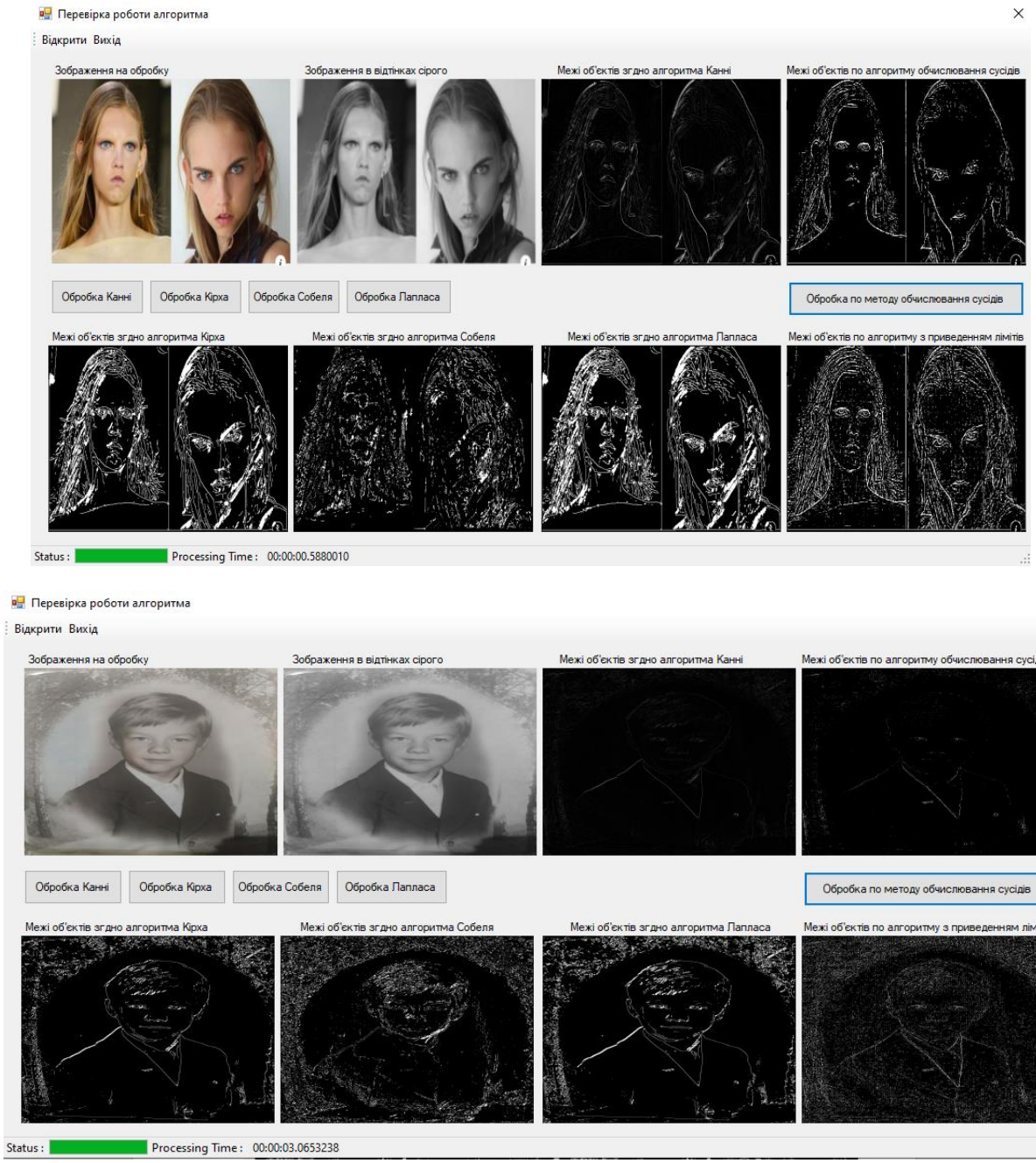



Рисунок 4.3 — Обробка фотографій

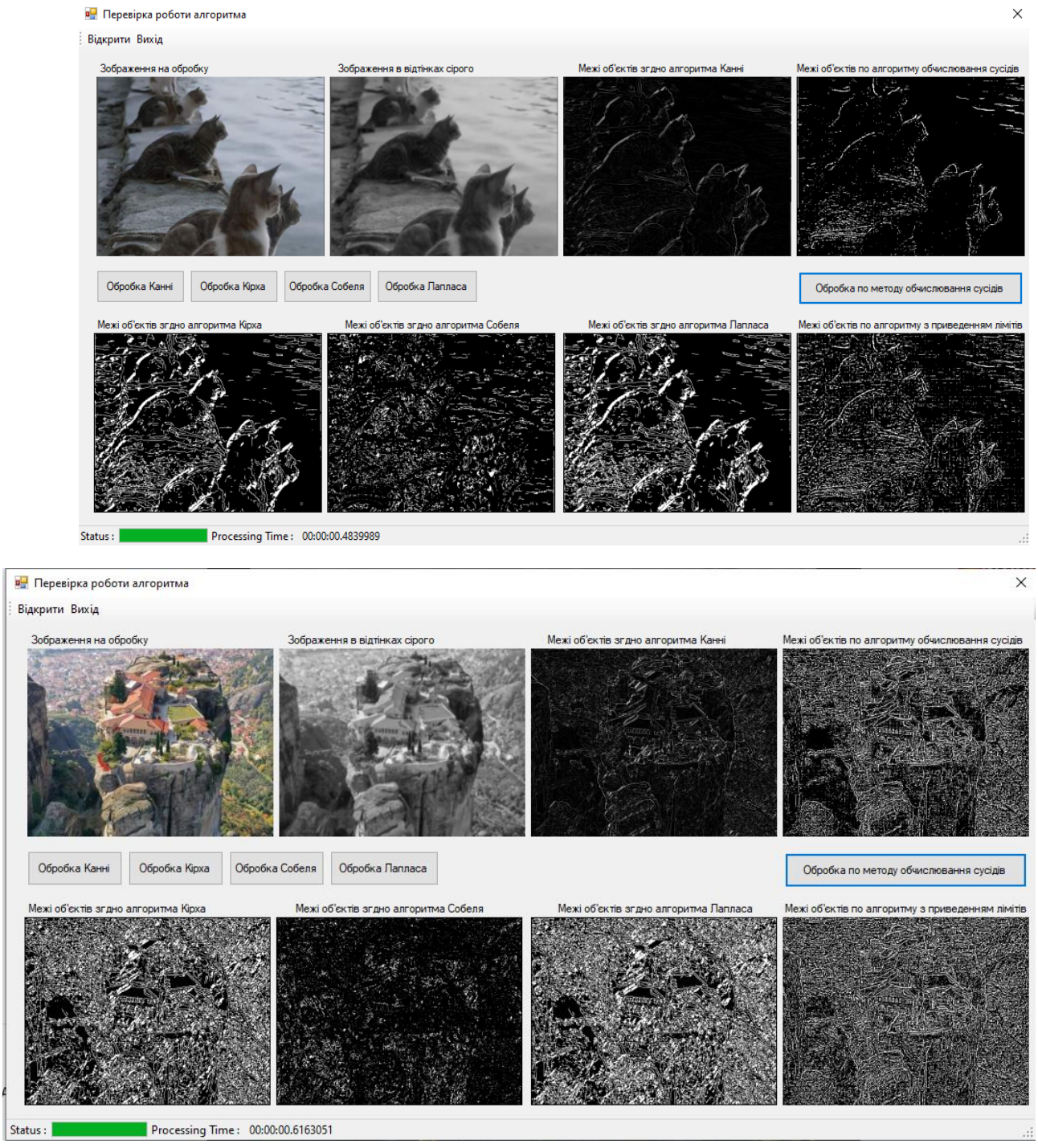



Рисунок 4.4 — Обробка панорам

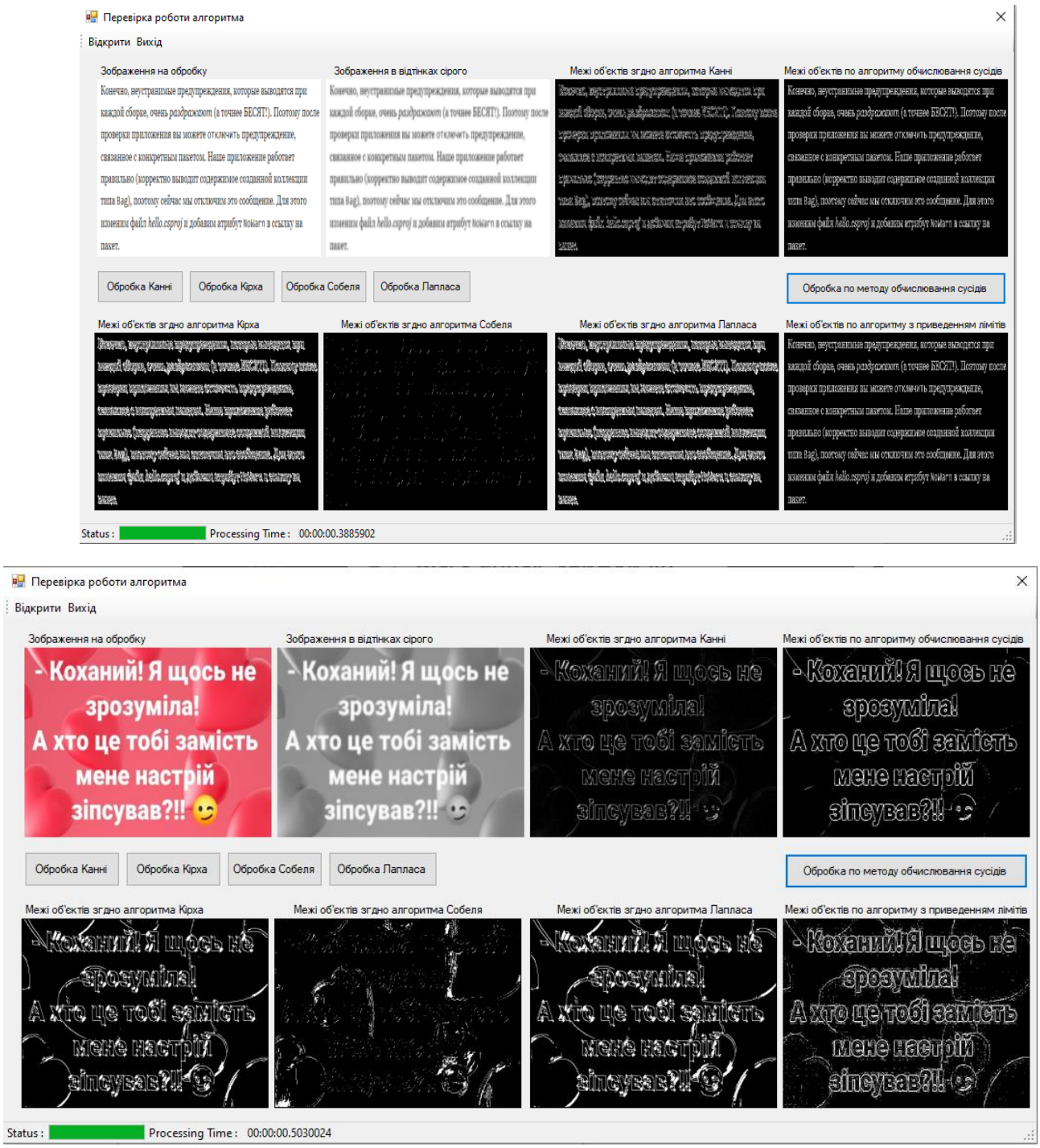






Рисунок 4.5 — Обробка тексту

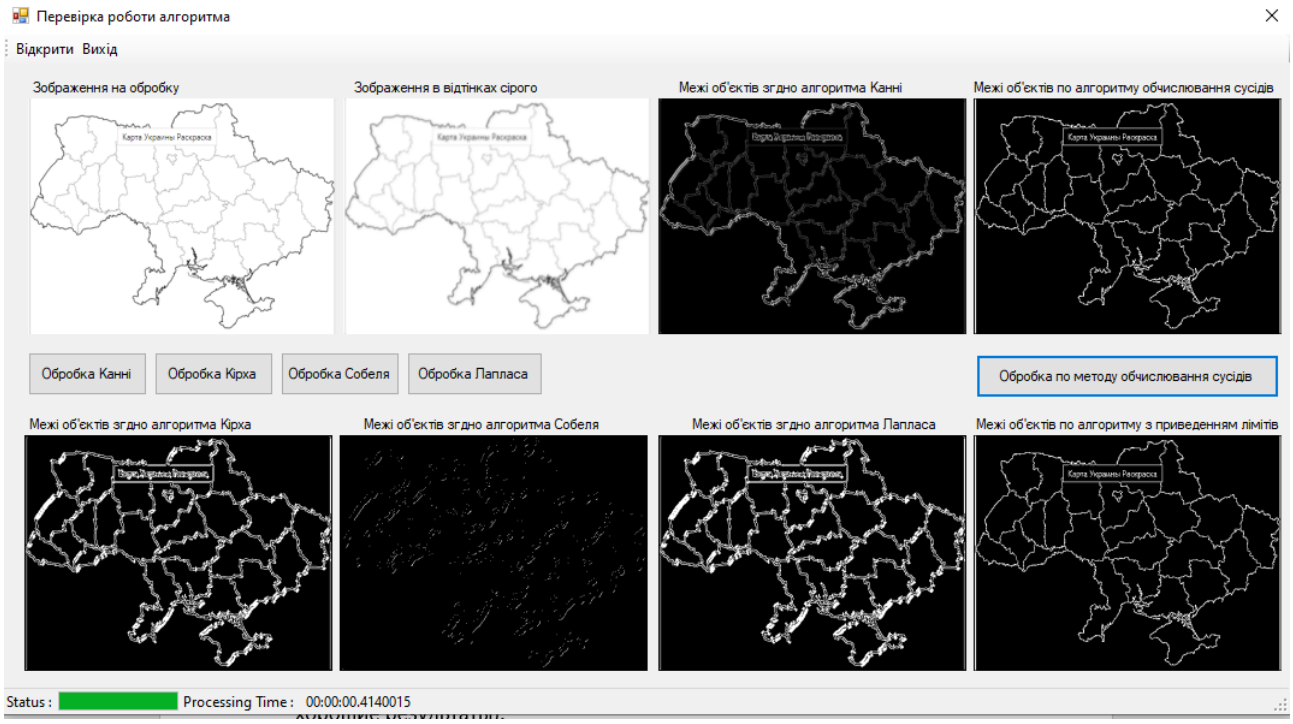



Рисунок 4.6 — Обробка контурів

Як видно з рисунків, у кожного алгоритму свої позитивні сторони та недоліки, залежно від вихідного зображення. Середньозважена оцінка критеріїв кожного з алгоритмів представлена у таблиці 4.1. Ієрархічний метод Оцу не розглядався порівняно, так як він застосовується до напівтонових зображень (при цьому на тестах показує добрі результати).

Таблиця 4.2 – Середньозважена оцінка критеріїв

| Алгоритм | Оцінка деталізації | Оцінка контурів | Оцінка якості |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| Модифікований алгоритм Канні пікселів із | 5.2 | 4.94 | 5.31 |

| | | | |
|---|---|---|---|
| застосуванням лімітів | | | |

Продовження таблиці 4.2

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| Модифікований алгоритм Канні | 6.16 | 5.71 | 4.70 |
| Оператор Канні | 5.25 | 5.1 | 5.03 |
| Оператор Кірша | 6.96 | 7.38 | 6.81 |
| Оператор Лапласа | 6.57 | 6.69 | 6.48 |

## 4.3 Середньозважена оцінка якості

Середньозважений бал оцінюваного критерію використовується як вхідні змінні системи нечіткого висновку для оцінки його якості. У цьому кожна змінна описується однією з трьох термів: П – «погано», З – «задовільно», Д – «добре». Для кожної з трьох змінних було встановлено функцію належності виду. Вихідна змінна "якість результатів виконання алгоритму" також описується одним із трьох зазначених термів. У таблиці 4.3 наведено деякі правила для визначення взаємозв'язків між вхідними та вихідними змінними.

Таблиця 4.3 - Правила взаємозв'язків між змінними

| **Деталізація** | **Контури** | **Якість зображення** | **Якість результатів алгоритмів** |
|---|---|---|---|

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| Д | Д | Д | Д |

Продовження таблиці 4.3

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| Д | З | Д | Д |
| З | Д | З | З |
| П | З | П | П |

## 4.4 Оцінка нечіткими множинами

Після проведення процедури дефазифікації з використанням методу центру тяжкості та застосування правил нечіткого висновку отримано такі результати (таблиця 4.4). З отриманих результатів можна зробити висновок, що за візуальною оцінкою найкращими є результати операторів Кірша та Лапласа. Ця перевага була віддана завдяки товщині отриманих ліній контуру та точності передачі деталей, необхідних для розуміння інформації, яку несе в собі зображення. Тому дуже дивно, що використання оператора Лапласа рідко згадується в інших роботах.

Таблиця 4.4 - Результати оцінки нечіткими множинами

| **Алгоритм** | **Якість результатів** |
|---|---|
| Модифікований алгоритм Канні пікселів із застосуванням лімітів | Гарна |
| Модифікований алгоритм Канні | Задовільна |
| Оператор Канні | Задовільна |
| Оператор Кірша | Задовільна |
| Оператор Лапласа | Гарна |

## 4.5 Оцінка обробки контурів

При отриманні контурів товщина кордону, що виділяється, повинна прагнути до однієї точки. Для оцінки ступеня здійсненності цього критерію у цій роботі впроваджується виділення контурів на тестовому зображенні (рис. 4.7). Кордони зображення окреслені одиничним контуром.

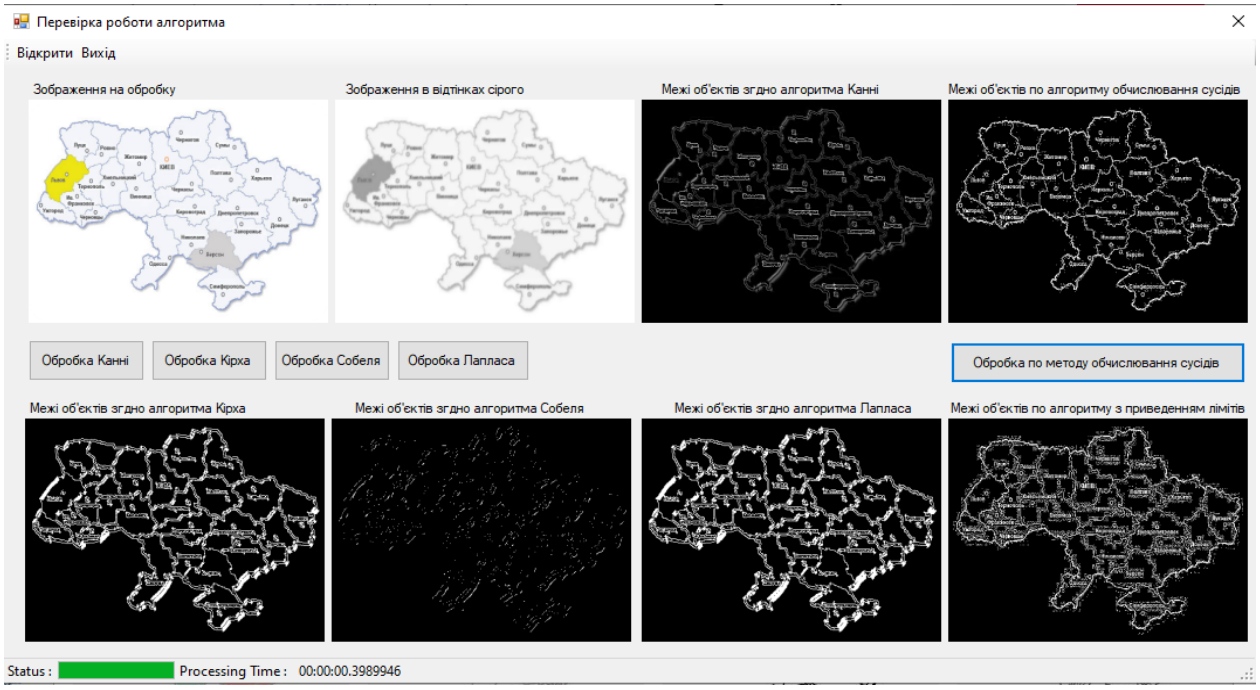



Рисунок 4.7 - Обробка контурів

З отриманих зображень без додаткових обчислень видно, що метод Канні та запропонований метод обробки сусідніх пікселів без застосування лімітів задовольняють цей критерій.

## 4.6 Чисельна оцінка якості виділення кордонів

Для чисельної оцінки якості виділення кордонів використовується метод на основі пікового відношення сигналу до шуму (PSNR) [27]. Для цього розраховуються три показники: вірогідність правильного детектування, ймовірність помилки першого роду, ймовірність помилки другого роду, використовуючи формули 3-5 відповідно. Тут кількість граничних точок в ідеальному контурі об'єкта визначається наступними параметрами: - кількість граничних точок в отриманому в процесі детектування контурі об'єкта; TP – кількість чітко визначених точок об'єкта; FN – кількість точок, які є граничними точками об'єкта, але які були виявлені детектором кордонів; FP – кількість точок, які є граничними точками об'єкта, але визначені детектором кордонів.

Таблиця 4.5 - Чисельна оцінка якості

| Метод | Оцінка якості | | | | |
|---|---|---|---|---|---|
| Еталон | 1 | 0 | 0 | 0.37 | 4.31 |
| Модифікований алгоритм Канні пікселів | 1 | 0 | 0 | 0.37 | 4.31 |
| Модифікований алгоритм Канні пікселів із застосуванням лімітів | 1 | 0 | 0 | 0.37 | 4.31 |
| Оператор Канні | 0.09 | 0.29 | 0.91 | 0.70 | 1.57 |
| Оператор Кірша | 0.28 | 0 | 0.72 | 0.60 | 2.21 |
| Лапласіан 5×5 | 0 | 0.20 | 1 | 0.72 | 1.42 |

## 4.7 Час виконання алгоритмів

Для оцінки часових витрат виконання алгоритмів було взято тестове зображення розмірністю 4140x5520 пікселів. Розрахунки проводилися на 2-х ядрах процесора Intel Core i7 з частотою 3.90 GHz. Результати тестування представлені у таблиці 4.6. Тестування показує, що найменш витратним ресурсом є метод POC, причому використання предиктора прискорює виконання основного алгоритму.

Таблиця 4.6 – Час виконання алгоритмів

| **Метод** | **Час виконання, мс** |
|---|---|
| Оператор Канні | 249 |
| Модифікований алгоритм Канні без застосування лімітів | 32 |
| Модифікований алгоритм Канні із застосуванням лімітів | 40 |
| Оператор Кірша | 70 |
| Оператор Лапласа | 55 |

## 4.8 Підвищення продуктивності – частковий перерахунок, багатопоточність у обробці

На думку автора, для підвищення продуктивності найкращими є два підходи:

- оптимізація внутрішніх обчислень, зменшення кількості операцій;
- використання багатопоточності.

Розглянемо обидва варіанти.

### Оптимізація внутрішніх обчислень

У розглянутому алгоритмі розрахунок ведеться кожного пікселя, починаючи з другого (i>=1 i<Wight, j>=1 j<Height), причому зчитуються всі сусідні пікселі (дев'ять значень кожного пікселя).

Якщо врахувати, що ми проводимо цю операцію дискретно і можемо заздалегідь знати, який піксель буде наступним, можна оновлювати не всі дев'ять осередків, а лише три з них. У цьому випадку операція пошуку коефіцієнта згладжування замість.

```
private int Cardinality(int[] s)
    {
       int k = 1;
       for (int i = 1; i < s.Length; i++)
       {
          if (s[i] != s[4])
             k++;
       }
       return k > 4? 4: k;
    }
```

Матиме вигляд:

```
private int Cardinality(int[] s)
    {
       int k = 1;
       for (int i = 1; i < s.Length; i++)
       {
          if (s[i]! = s[i-1])
             k++;
          if (k == 4)
             return k;
```

```
        }
        return k;
    }
```

Також для кожної з ітерацій слід уточнювати, який із елементів буде «середнім» для обчислень. У таблиці 4.7 продемонстровано, як змінюватимуться значення масиву, що обробляється під час проходження матриці зображення горизонталлю.

Таблиця 4.7 – Демонстрація масиву

| Номер пікселя по горизонталі (i) | Замінні значення масиву Normal | Значення лічильника | Замінні значення масиву Optimal | Візуальне подання досліджуваних осередків |
|---|---|---|---|---|
| 1 | 1...9 | 1 | 1..9 | * * *<br>* * *<br>* * * |
| 2 | 1...9 | 2 | {4,5,6} | *<br>*<br>* |
| 3 | 1...9 | 3 | {7,8,9} | *<br>*<br>* |
| 4 | 1...9 | 1 | {1,2,3} | *<br>*<br>* |
| 5 | 1...9 | 2 | {4,5,6} | *<br>*<br>* |
| | | | | |
| Wight-1 | 1...9 | 1 \|\| 2 \|\| 3 | {1,2,3} | *<br>*<br>* |

Як очевидно з таблиці, ми на 30-60 відсотків зменшуємо кількість операцій присвоєння. Слід врахувати, що чим більша величина зображення, тим явніше збільшення продуктивності спостерігатиметься.

Крім того, потрібно окремо вказати, що дане прискорення ми не вносимо в опис власне алгоритму у зв'язку з тим, що для пояснення суті алгоритму на даному етапі важливо скоріше розуміння його роботи, ніж гора формул, за якою може загубитися сам зміст алгоритму.

## 4.9 Використання багатопоточності

Сучасна архітектура обчислювальних потужностей дозволяє розпаралелити обчислення, запустивши кілька потоком і, відповідно, асинхронно обчислювати значення окремих елементів матриці пікселів.

Можливо два шляхи розпаралелювання завдань:

- запуск асинхронної обробки по кожному ряду вхідної матриці I[I,j] – у цьому випадку ми отримуємо j окремих потоків, які заповнюють кожен свій ряд результуючої матриці.

Лістинг такого рішення:

```
public void CalculateMatrix()
        {
           Thread [] t = New Thread [Hight];

           for (int i = 0; i < Hight; i++)
           {
              t[i] = New Thread(New ParameterizedThreadStart(RunCalcRow));
              t[i].Start(i);
              t [i]. Join (i);
              //Console.WriteLine("Основний потік: Завершення робочого потоку. {0}",i);
              //Console.WriteLine("Час виконання: {0}", ts.TotalMilliseconds);
           }
        }
```

- запуск асинхронної обробки по кількох областях матриці в деяку (заздалегідь визначену) кількість потоків.

У цьому випадку матрицю I[I,j] розділяємо на (наприклад) чотири області, кожна з яких здійснює обчислення в окремому потоці, щоб потім записати отримані результати в результуючу матрицю.

На практиці за наявності більш-менш сучасного процесора ми не домагаємося прискорити роботу алгоритму, а швидше навпаки, час на створення та подальший супровід паралельних обчислень перевищує час синхронного виконання операцій обчислення в одному потоці.

Застосування асинхронних операцій поділу потоків виконання, на думку автора проєкту, може виявитися корисним при обробці відеосигналу, в цьому випадку ми відразу запускаємо обробку кожного з кадрів відеопотоку партіями по три-чотири потоки, і поки виводяться результат виконання першої операції (першого кадру), операції Обробки виконуються над другим, третім кадром.

# Висновки до четвертого розділу

У четвертому розділі продемонстровано роботу модифікованого алгоритма. Розглянуті оцінки критеріїв порівняння. Модифікований алгоритм порівнюється з алгоритмом Канні та представлено що саме було покращенно. Також порівнюється з такими алгоритмами, як оператор Собеля, оператор Кірша та оператор Лапласа. За такими оцінками якості: швидкість, якість та затрата пам’яті.

Було визначено, що запропонований модифікований алгоритм працює краще за сам Алгоритм Канні по параметрам: виділення контуру та швидкості. Також він працює краще за деякі вже існуючі алгоритми за всіма параметрами. Але варто сказати, що оператор Собеля краще виділяє межі об’єктів, але працює значно гірше.

# ВИСНОВКИ

Ця магістерська робота присвячена модифікації алгоритму Канні для знаходження меж об’єктів зображення. В ході роботи було розглянуто актуальність теми. Проаналізовано існуючі алгоритми для обробки картинок. Запропоновано покращення алгоритму Канні.

Провела дослідження в обробці зображення, визначила завдання, які повині виконувати алгоритми та склала методи покращення алгоритму Канні.

Описані всі кроки модифікованого алгоритму. Представлені в вигляді коду, як працює алгоритм.

За допомогою програми виведено та продемонстровано порівняння з існуючими алгоритмами. Представлені переваги та чим краще запропонований алгоритм. Визначила слабкі точки та де ще можна покращити роботу.

Модифікований алгоритм реалізує заздалегідь визначені функції: виділення меж об’єктів на зображенні.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Дж. Лі, Б. Уєр. Тривимірна графіка та анімація. - 2-ге вид. - М.: Вільямс, 2002. 640 с.
2. Дональд Херн, М. Паулін Бейкер. Комп'ютерна графіка та стандарт OpenGL = Computer Graphics with OpenGL. - 3-тє вид. - М.: "Вільямс", 2005. 1168 с.
3. Едвард Енджел. Інтерактивна комп'ютерна графіка Вступний курс з урахуванням OpenGL = Interactive Computer Graphics. A Top-Down Approach with Open GL. - 2-ге вид. - М.: "Вільямс", 2001. 592 с.
4. Енджел Е. Інтерактивна комп'ютерна графіка. Вступний курс на базі OpenGL. - 2-ге вид. - М.: Вільямс, 2001. 592 с.
5. Іванов В. П., Батраков А. С. Тривимірна комп'ютерна графіка. Під ред. Г. М. Поліщука. - М.: Радіо та зв'язок, 1995. 224 с.
6. Інженерна та комп'ютерна графіка : підруч. для студ. ВНЗ/В.Є. Михайленко, В. В. Ванін, С. М. Ковалев. - К.: Каравела, 2010. 360 с.
7. Комп'ютер малює фантастичні світи (ч.2). Комп'ютер знаходить розум = Artificial Intelligence Computer Images. / За ред. В.Л. Стефанюка. - М.: Світ, 1990. - 240 с.
8. Конушин А. Слежение за точечными особенностями сцены (Point feature tracking). Компьютерная графика и мультимедиа. Выпуск №1(5)/2003.
9. Нікулін Є. А. Комп'ютерна геометрія та алгоритми машинної графіки. - СПб.: БХВ-Петербург, 2003. 560 с.
10. Нікулін Є.А. Комп'ютерна графіка. Моделі та алгоритми (недоступне посилання). СПб: видавництво "Лань". – 708 с. (2017). Дата звернення: 24 листопада 2018 року. Архівовано 24 листопада 2018 року.
11. Снук Г.. 3D-ландшафти в реальному часі на C++ і DirectX 9. - 2-е вигляд. - М.: Кудіц-прес, 2007. 368 с.
12. Херн Д., Бейкер М. П. Комп'ютерна графіка та стандарт OpenGL. - 3-тє вигляд. - М.: Вільямс, 2005. 1168 с.
13. Abhishak Yadav, Poonam Yadav. Digital Image Processing, 2009.

14. Accurate Junction Detection and Characterization in Natural.
15. Andres Solis Montero, Milos Stojmenović, Amiya Nayak. Robust Detection of Corners and Corner-line links in images. 10th IEEE International Conference on Computer and Information Technology (CIT 2010), 2010.
16. Awrangjeb M.. CPDA, 2009.
17. Bae S.C., Kweon I.S. and Yoo C.D. COP: a new corner detector. Pattern Recogn. Lett., 2002.
18. Drummond T. Fusing Points and Lines for High Performance Tracking, 2005.
19. Chen He X., Yung N.. Corner detector based on global and local curvature, 2008.
20. Edward Rosten. Faster and better: a machine learning approach to corner detection, 2008.
21. Farzin Mokhtarian. Robust Image Corner Detection Through Curvature Scale Space, 1998.
22. Harris Corner Detector.
23. Kahaki S. M. M., Contour-Based Corner Detection and Classification by Using Mean Projection Transform, 2014.
24. Maha Nakhon, Thailand's Tallest Building, Completes Архівна копія від 3 червня 2017 року на Wayback Machine (англ.) на сайті ctbuh.org, 4 травня 2020.
25. Maha Nakhon, la tour pixellisee de Bangkok Архівна копія від 15 жовтня 2017 року на Wayback Machine (фр.) на сайті wandererz.net, 20 жовтня 2020
26. Miroslav M. H. Trajkovii. Fast corner detection, 1998.
27. Otsu N. - A threshold selection method from gray-level histogram (IEEE Trans. Syst. Man Cybern. 9:62-66;1979.
28. Pixel — Definition and More from the Free Merriam-Webster Dictionary
29. Rattarangsi A., W. U. M. W. U. Dept. of Electr. & Comput. Eng. и R. Chin, Scale-based detection of corners of planar curves, Pattern Analysis and Machine Intelligence, IEEE Transactions on (Volume:14, Issue: 4), 1992.
30. Rodehorst V., Koschan A. Comparison and evaluation of feature point detectors, 2006.

31. Shi T.. Good Features to Track, 1994.
32. Tinne Tuytelaars, Krystian Mikolajczyk. Local Invariant Feature Detectors: A Survey, 2008.
33. Zamofing T., Hugli H., Applied multifocus 3D microscopy. Proceedings of SPIE. 2004, pp. 134-144.
34. http://www.cs.toronto.edu/~jepson/csc420/notes/imageFeaturesIIBinder.pdf.