Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

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

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

ДОСЛІДЖЕННЯ ТА РЕАЛІЗАЦІЯ МЕТОДІВ ФІЛЬТРАЦІЇ ТА ЗНАХОДЖЕННЯ КОНТУРУ ОБ'ЄКТА
(тема)

Виконав:
студент 2 курсу, групи ІНФм-19-2
Монастирський В.Р.
(прізвище, ініціали)

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

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Любченко В.А.
(посада, прізвище, ініціали)

Допускається до захисту

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

2020 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти другий (магістерський)

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

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 20 ___ р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

студентові Монастирському Владиславу Романовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження та реалізація методів фільтрації та знаходження контуру об'єкта

затверджена наказом по університету від «23» жовтня 2020 року № 1428Ст.

2. Термін подання студентом роботи до екзаменаційної комісії 2 грудня 2020 р.

3. Вихідні дані до роботи
Алгоритми аналізу контуру зображень
Тестові зображення програмної реалізації
Теоретичні відомості про оператори аналізу контуру зображень

4. Перелік питань, що потрібно опрацювати в роботі
1. Огляд алгоритмів детектування контурів зображення;
2. Аналіз алгоритмів детектування контурів зображення;
3. Актуальність розпізнавання образів у сучасному житті;
4. Математична модель детектування контурів зображення;
5. Реалізація алгоритму аналізу контурів на базі операторів Робертса, Превітта, Собеля, Шарра, Кірша та Робінсона;
6. Розробка WEB інтерфейсу;

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) _____

1. Тестові зображення

2. Математичні формули

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

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на атестаційну роботу	07.09.20-08.09.20	
2	Аналіз завдання, підбір літератури	10.09.20-24.09.20	
3	Аналіз літератури з досліджуваної проблеми	25.09.20-26.09.20	
4	Аналіз технічних засобів	28.09.20-01.10.20	
5	Модифікація алгоритму ДК	30.09.20-02.10.20	
6	Програмна реалізація ІК	03.10.20-15.10.20	
7	Оформлення пояснювальної записки	16.11.20-17.11.20	
8	Перевірка на плагіат	17.11.20-21.11.20	
9	Рецензування	23.11.20-05.12.20	
10	Підготовка презентації та доповіді	03.12.20-08.12.20	
11	Занесення роботи в електронний архів	09.12.20	
12	Попередній захист атестаційної роботи	09.12.20	

Дата видачі завдання 23 жовтня 2020 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Любченко В.А.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до атестаційної роботи: 61 с., 1 табл., 22 рис., 1 дод., 40 джерел.

ОПЕРАТОР РОБЕРТСА, ОПЕРАТОР ПРЕВІТТА, ОПЕРАТОР СОБЕЛЯ, ОПЕРАТОР ШАРРА, ОПЕРАТОР КІРША, ОПЕРАТОР РОБІНСОНА, ІДЕНТИФІКАЦІЯ КРАЇВ, ДЕТЕКТУВАННЯ КОНТУРІВ, РОЗПІЗНАВАННЯ ОБРАЗІВ, ФІЛЬТР, МАСКА, ЯДРО.

Робота присвячена вирішенню проблеми розпізнавання образів на зображеннях. Було реалізовано виділення контурів об’єктів на зображенні за допомогою алгоритмів виділення контурів. Ці методи грунтуються на одній з базових властивостей сигналу яскравості – розривність. Пошук розривів відбувається за допомогою ковзної маски, також відомої як фільтр або ядро, яка являє собою квадратну матрицю відповідну до вказаної групи пікселів вихідного зображення. Елементи маски також називають коефіцієнтами. Використання цієї маски в яких небудь локальних перетвореннях називають просторовою фільтрацією або просто фільтрацією. Тобто процес фільтрації заснований на переміщенні маски фільтра від точки до точки зображення.

У результаті роботи здійснена програмна реалізація системи для детектування контурів об’єктів на зображенні.

ROBERTS OPERATOR, PREWITT OPERATOR, SOBEL OPERATOR, SHARR OPERATOR, KIRSH OPERATOR, ROBINSON OPERATOR, EDGE IDENTIFICATION, DETECTION OF CONTOURS, PATTERN RECOGNITION, FILTER, MASK, CORE.

The work is devoted to solving the problem of image recognition in images. The contours of objects in the image were selected using contour selection algorithms. These methods are based on one of the basic properties of the luminance signal - discontinuity. The gaps are searched using a sliding mask, also known as a filter or core, which is a square matrix corresponding to the specified group of pixels of the original image. Mask elements are also called coefficients. The use of this mask in any local transformations is called spatial filtering or simply filtering. That is, the filtering process is based on moving the filter mask from point to point in the image.

As a result, the software implementation of the system for detecting the contours of objects in the image was implemented.

ЗМІСТ

Перелік умовних позначень, символів, одиниць, скорочень і термінів 7
Вступ 8
1 Детектування контурів 9
1.1 Ціль розпізнавання контурів об’єктів на зображенні 9
1.2 Властивості алгоритмів ідентифікації контурів 10
1.3 Проста модель детектування контурів 11
1.4 Чому це не тривіальна задача 11
1.5 Підходи до вирішення задачі 13
1.5.1 Методи першого порядку 13
1.5.2 Порогове обмеження та зв’язування 15
1.5.3 Витончення контурів 16
1.5.4 Методи другого порядку 17
1.5.5 Диференціальні методи 17
1.5.6 Фазова когерентність 19
1.5.7 Базування на фізиці 20
1.5.8 Субпіксельний підхід 21
2 Огляд основних методів детектування контурів об’єктів 22
2.1 Техніки ідентифікації контурів 22
2.2 Оператор Робертса 22
2.3 Оператор Превітта 23
2.4 Оператор Собеля 24
2.5 Оператор Шарра 25
2.6 Оператор Кірша 27
2.7 Оператор Робінсона. 28
2.8 Постановка задачі дослідження 29
3 Математичні моделі операторів детектування контурів об’єктів 30
3.1 Оператор Робертса 30
3.2 Оператор Превітта 31

3.3 Оператор Собеля 33
3.4 Оператор Шарра 36
3.5 Оператор Кірша 38
3.6 Оператор Робінсона 39
4 Комп’ютерна модель детектування контурів 41
4.1 Обґрунтування вибору середовища програмної реалізації 41
4.1.1 IDE WebStorm 41
4.1.2 JavaScript 41
4.1.3 PIXI.js 43
4.1.4 React.js 43
4.2 Програмна реалізація 44
4.3 Інструкція користувача 48
4.4 Тестування розробленої моделі 49
4.4.1 Тестування програми 49
4.4.2 Аналіз роботи програми 54
Висновки 56
Перелік джерел посилання 57
Додаток А Тестові зображення 61

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ДК – детектування контурів

ІК – ідентифікація країв

N – North (Північ)

NW – North West (Північний Захід)

W – West (Захід)

SW – South West (Південний Захід)

S – South (Південь)

SE – South East (Південний Схід)

E – East (Схід)

NE – North East (Північний Схід)

JS – JavaScript

РО – розпізнавання образів

ВСТУП

Обробка зображень – це метод аналізу та обробки цифрових зображень за допомогою математичних операторів. При обробці зображення вхідними даними є зображення і результат може бути або набором характеристик, або набором параметра зображення або зображення. Зображення містить різну інформацію, таку як контур об’єкта, його орієнтація, розмір і колір. Отже, щоб знайти інформацію про форму об’єкта, ребра, що входять в цей об’єкт повинен бути ідентифікований.

Виявлення країв – це метод виявлення виникнення країв та його локальність, яка створюється різкими варіаціями інтенсивності (яскравість або колір) зображення. Розриви зображення можуть відрізнятися від освітленості сцени, орієнтація поверхні, її глибини та варіацій властивостей матеріалів. Завдання виявлення країв – це виявлення інформації про фігуру об’єкта. Це важливий крок в аналізі та обробці зображень, комп’ютерному зорі, людському зорі, виявленні об’єктів та розпізнаванні образів. Існують різні методи виявлення контурів.

Контурні детектори працюють по-різному. Це означає, що деякі крайові детектори займають більше часу і виявляють більше країв по відношенню до інших. Виявлення контурів на зображенні залежить від інтенсивності, освітленості, шуму та розмитості.

У цій роботі вивчаються та порівнюються різні методи виявлення контурів для ідентифікації країв об’єктів на зображенні.

1 ДЕТЕКТУВАННЯ КОНТУРІВ

1.1 Ціль розпізнавання контурів об’єктів на зображенні

Метою виявлення різких змін яскравості зображення є фіксація важливих подій та змін властивостей світу. Можна показати, що за досить загальних припущень для моделі формування зображення розриви в яскравості зображення, ймовірно, відповідають:

- глибокі розриви;
- розриви в орієнтації поверхні;
- зміни властивостей матеріалу;
- варіації освітленості сцени.

В ідеальному випадку результат застосування крайового детектора до зображення може призвести до набору з’єднаних кривих, які вказують межі об’єктів, межі поверхневих розміток, а також криві, що відповідають розривам в орієнтації поверхні. Таким чином, застосування алгоритму виявлення країв до зображення може значно зменшити обсяг даних, що підлягають обробці, і можна відфільтрувати інформацію, яка може вважатися менш актуальною, зберігаючи при цьому важливі структурні властивості зображення. Якщо крок виявлення краю вдалий, подальше завдання інтерпретації інформаційного вмісту на вихідному зображенні може бути істотно спрощеним. Однак отримати такі ідеальні краї не завжди вдається із реальних образів помірної складності.

Краям, які отримані з зображень, часто заважає фрагментація, що означає, що крайові криві не з’єднані, відсутні сегменти ребер, а також помилкові ребра, що не відповідають цікавим явищам на зображенні, що ускладнює подальше завдання інтерпретації даних зображення.

Виявлення країв є одним з основних етапів у обробці зображень, аналізі зображень, розпізнаванні образів зображень та техніках комп’ютерного зору (рис. 1.1).

Рисунок 1.1 – Приклад ідентифікації контурів

1.2 Властивості алгоритмів ідентифікації контурів

Краї, виділені з двовимірного зображення тривимірної сцени, можна класифікувати як залежні від точки зору, так і незалежні від точки зору. Незалежний край точки зору зазвичай відображає властиві тривимірних об’єктів, такі як форма поверхні. Залежний край точки зору може змінюватися в міру зміни точки зору, і, як правило, відображає геометрію сцени, наприклад об’єкти, які закривають один одного.

Типовим краєм може бути, межа між блоком червоного кольору та блоком жовтого. На відміну від цього, лінія може мати невелику кількість пікселів іншого кольору на іншому незмінному тлі. Тому для лінії, як правило, може бути один край з кожного боку лінії.

1.3 Проста модель детектування контурів

Хоча розглядається питання виявлення ідеальних контурів, краї отримані від природних зображень, як правило, зовсім не є ідеальними. Натомість на них зазвичай впливає один або декілька з наступних ефектів:

– фокусна розмитість, спричинена кінцевою глибиною різкості та функцією розподілу кінцевих точок;

– розмиття півтіні, викликане тінями, створеними джерелами світла ненульового радіуса;

– затінення на гладкому об’єкті.

Ряд дослідників використовували згладжений по Гаусу крок (функція помилки) як найпростіше розширення ідеальної моделі кроку ребра для моделювання ефектів розмиття країв у практичному застосуванні. Таким чином, одновимірне зображення *f*, яке має рівно один край, розміщений у $x = 0$, може бути змодельований як [1]:

$$\boldsymbol{f(x) = \frac{I_r - I_l}{2}\left(erf\left(\frac{x}{\sqrt{2\sigma}}\right) + 1\right) + I_l.} \tag{1.1}$$

Ліворуч від краю інтенсивність дорівнює $I_l = \lim_{x\to-\infty} f(x)$, а праворуч від краю – $I_r = \lim_{x\to\infty} f(x)$. Параметр масштабу σ називається шкалою розмиття краю. В ідеалі цей параметр масштабу слід регулювати залежно від якості зображення, щоб уникнути руйнування справжніх країв зображення.

1.4 Чому це не тривіальна задача

Щоб проілюструвати, чому виявлення країв не є тривіальним завданням, розглянемо проблему виявлення країв у наступному

одновимірному сигналі. Тут можна інтуїтивно сказати, що між 4-м і 5-м пікселями повинен бути край [2] (рис. 1.2).

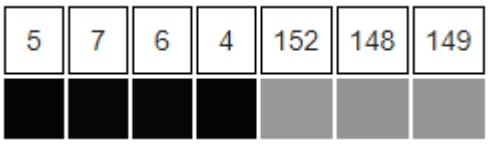

Рисунок 1.2 – Одновимірний сигнал

Якби різниця інтенсивності була меншою між 4-м і 5-м пікселями і якщо різниця інтенсивності між сусідніми пікселями була би більшою, то було б не так просто сказати, що у відповідному регіоні повинен бути край. Більше того, можна стверджувати, що це послідовність, в якій є декілька країв [3] (рис. 1.3).

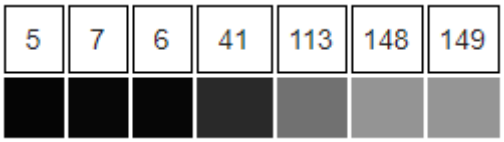

Рисунок 1.3 – Одновимірний сигнал

Отже, твердо визначити конкретний поріг того, наскільки великою може бути зміна інтенсивності між двома сусідніми пікселями, щоб можна було сказати, що між цими пікселями має бути край, не завжди просто. Дійсно, це одна з причин, чому виявлення країв може бути нетривіальною проблемою, якщо об’єкти на сцені особливо прості і умови освітлення можна добре контролювати.

1.5 Підходи до вирішення задачі

Існує безліч методів виявлення країв, але більшість із них можна згрупувати у дві категорії – на основі пошуку та переходу через нуль. Методи, засновані на пошуку, виявляють ребра, спочатку обчислюючи міру сили ребра, як правило, похідний вираз першого порядку, такий як величина градієнта, а потім здійснюючи пошук локальних спрямованих максимумів величини градієнта, використовуючи обчислювальну оцінку локальної орієнтації краю, як правило, напрямок градієнта [4]. Методи, що базуються на нульовому перетині, шукають нульові перехрещення у похідному виразі другого порядку, обчисленому на основі зображення, з метою пошуку ребер, як правило, нульових перетинів Лапласіана чи нульових перетинів нелінійного диференціального виразу. В якості етапу попередньої обробки до виявлення краю майже завжди застосовується етап згладжування, як правило, згладжування за Гаусом.

Опубліковані методи виявлення країв в основному відрізняються типом застосовуваних згладжувальних фільтрів та способом обчислення міри міцності краю. Оскільки багато методів виявлення країв покладаються на обчислення градієнтів зображення, вони також відрізняються типами фільтрів, що використовуються для обчислення оцінок градієнтів у напрямках x та y.

1.5.1 Методи першого порядку

Для оцінки градієнтів зображення з вхідного зображення або згладженої його версії можуть застосовуватися різні оператори градієнта. Найпростіший підхід полягає у використанні центральних відмінностей:

$$\boldsymbol{L_x(x,y) = -\frac{1}{2}L(x-1,y) + 0 * L(x,y) + \frac{1}{2} * L(x+1,y),}$$
$$\boldsymbol{L_y(x,y) = -\frac{1}{2}L(x,y-1) + 0 * L(x,y) + \frac{1}{2} * L(x,y+1).} \quad (1.2)$$

що відповідає застосуванню наступних масок фільтру до даних зображення:

$$\boldsymbol{L_x = \begin{bmatrix} \frac{1}{2} & 0 & -\frac{1}{2} \end{bmatrix} L \quad and \quad L_y = \begin{bmatrix} \frac{1}{2} \\ 0 \\ -\frac{1}{2} \end{bmatrix} L.} \quad (1.3)$$

Відомий і раніше відомий оператор Собеля базується на таких фільтрах:

$$\boldsymbol{L_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} L \quad and \quad L_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} L.} \quad (1.4)$$

Враховуючи такі оцінки похідних зображень першого порядку, величина градієнта потім обчислюється як:

$$\boldsymbol{|\nabla L| = \sqrt{L_x^2 + L_y^2}.} \quad (1.5)$$

Тоді як орієнтацію градієнта можна оцінити як

$$\boldsymbol{\theta = atan2(L_x, L_y).} \quad (1.6)$$

Інші оператори різниці першого порядку для оцінки градієнта зображення були запропоновані в операторі Превітта, Робертса Кроса, оператора Каялі та оператора Фрей-Чена.

Можна розширити розмір фільтрів, щоб уникнути проблеми розпізнавання краю на зображенні з низьким SNR. Вартість цієї операції – втрата з точки зору роздільної здатності.

1.5.2 Порогове обмеження та зв’язування

Після того, обчисливши міру сили краю (як правило, величина градієнта), наступним етапом є застосування порога, щоб вирішити, чи є краї в точці зображення чи ні. Чим нижчий поріг, тим більше країв буде виявлено, а результат буде дедалі більш сприйнятливим до шуму та виявлення країв невідповідних об’єктів на зображенні. І навпаки, високий поріг може пропускати тонкі краї або призвести до фрагментованих країв.

Якщо край застосовувати лише до зображення з градієнтною величиною, то отримані краї, як правило, будуть товстими, і необхідна якась постійна обробка краю. Однак для ребер, виявлених із придушенням не максимуму, криві ребра тонкі за визначенням, а пікселі ребер можуть бути зв’язані в полігон ребра процедурою зв’язування ребер (відстеження ребер). На дискретній сітці етап не максимуму придушення може бути реалізований шляхом оцінки напрямку градієнта за допомогою похідних першого порядку, потім округлення напрямку градієнта до кратних 45 градусів і, нарешті, порівняння значень величини градієнта в розрахунковому градієнті напрямку.

Загальновживаний підхід для вирішення проблеми відповідних порогових значень для порогового значення – це використання порогових значень з гістерезисом. Цей метод використовує кілька порогів для пошуку країв. Починається з використання верхнього порогу, щоб знайти початок ребра. Отримавши початкову точку, треба простежити шлях краю через піксель зображення за пікселем, позначаючи край, коли перебуваємо вище нижнього порогу [5]. Припинення маркування краю лише тоді, коли значення

опускається нижче нижнього порогу. Цей підхід робить припущення, що краї, ймовірно, будуть у суцільних кривих, і дозволяє нам слідувати по слабкій ділянці ребра, яку ми бачили раніше, але це не означає, що кожен шумний піксель на зображенні позначений як край. Однак, існує проблема вибору відповідних параметрів порогового значення, і відповідні порогові значення можуть відрізнятися залежно від зображення.

1.5.3 Витончення контурів

Витончення контурів – це техніка, яка використовується для видалення небажаних фальшивих точок по краях зображення. Ця техніка застосовується після того, як зображення відфільтровано на шум (із використанням медіани, фільтра Гауса тощо), застосовано оператор краю (як описані вище, Canny або Sobel) для виявлення країв і після згладжування країв використовуючи відповідне порогове значення. Це видаляє всі небажані точки і при обережному застосуванні створює крайові елементи товщиною в один піксель.

Переваги цього підходу:

– гострі та тонкі краї призводять до більшої ефективності розпізнавання об’єктів;

– якщо для виявлення ліній та еліпсів використовують перетворення Хафа, то стоншення може дати набагато кращі результати;

– якщо край виявляється межею області, то витончення може легко дати параметри зображення, такі як периметр, без особливої алгебри.

Для цього використовується багато популярних алгоритмів, один із них описаний нижче:

– треба обрати тип підключення, наприклад, 8, 6 або 4;

– 8 є кращим зв’язком, коли розглядаються всі безпосередні пікселі, що оточують певний піксель;

– видалити точки з півночі, півдня, сходу та заходу;

– робити це за кілька циклів, тобто після північного проходу використовуйте одне і те ж напівоброблене зображення в інших проходах тощо;

– кількість проходів через напрямок слід вибирати відповідно до бажаного рівня точності.

1.5.4 Методи другого порядку

Деякі оператори виявлення краю натомість базуються на похідних другого порядку інтенсивності. Це, по суті, фіксує швидкість зміни градієнта інтенсивності. Таким чином, в ідеальному неперервному випадку виявлення перетину нуля у другій похідній фіксує локальні максимуми в градієнті.

Ранній оператор Марра-Гілдрета заснований на виявленні перетину нуля оператора Лапласіана, застосованого до згладженого за Гаусом зображення. Однак можна показати, що цей оператор також поверне помилкові ребра, що відповідають локальним мінімумам величини градієнта [6]. Більше того, цей оператор дасть погану локалізацію на вигнутих краях. Отже, сьогодні цей оператор представляє переважно історичний інтерес.

1.5.5 Диференціальні методи

Більш вдосконалений підхід до виявлення ребер другого порядку, який автоматично виявляє ребра з точністю до пікселів, використовує наступний диференціальний підхід для виявлення нульових перетинів похідної спрямованості другого порядку у напрямку градієнта.

Дотримуючись диференціально-геометричного способу вираження вимоги не максимального придушення, запропонованого Ліндебергом,

введемо в кожній точці зображення локальну систему координат *(u, v)*, з напрямком *v*, паралельним напрямку градієнта. Якщо припустити, що зображення було попередньо згладжено за допомогою згладжування Гауса та обчислено масштабне представлення простору *L(x,y;t)* у масштабі *t*, то можна вимагати, щоб величина градієнта подання просторової шкали, яка дорівнює похідній напрямку першого порядку у *v*-напрямку напряму L_v, мала перший порядок похідної у напрямку *v* [7], яка рівна нулю:

$$\boldsymbol{\vartheta_v(L_v) = 0}, \tag{1.7}$$

в той час як похідна спрямованості другого порядку у *v*-напрямку L_v повинна бути від'ємною, тобто:

$$\boldsymbol{\vartheta_{vv}(L_v) \leq 0}. \tag{1.8}$$

Виписано як явний вираз з точки зору локальних часткових похідних L_x, L_y, … , L_{yyy}, це визначення краю можна виразити як криві перетину нуля диференціального інваріанта:

$$\boldsymbol{L_v^2 L_{vv} = L_x^2 L_{xx} + 2L_x L_y L_{xy} + L_y^2 L_{yy} = 0}, \tag{1.9}$$

які задовольняють знакову умову на наступному диференціальному інваріанті:

$$\boldsymbol{L_v^3 L_{vvv} = L_x^3 L_{xxx} + 3L_x^2 L_y L_{xxy} + 3L_x L_y^2 L_{xyy} + L_y^3 L_{yyy} \leq 0}. \tag{1.10}$$

Позначають часткові похідні, обчислені з представлення масштабного простору *L*, отриманого згладжуванням вихідного зображення гауссовим ядром. Таким чином, краї будуть автоматично отримані як безперервні криві

з точністю до пікселів. Порогове значення гістерезису також може застосовуватися до цих диференціальних та субпіксельних крайових сегментів.

На практиці наближення похідних першого порядку можна обчислити за допомогою центральних різниць, як описано вище, тоді як похідні другого порядку можна обчислити із представлення просторового масштабу L згідно:

$$\begin{aligned} \boldsymbol{L_{xx}(x,y) = L(x-1,y) - 2L(x,y) + L(x+1,y)}, \\ \boldsymbol{L_{xy}(x,y) = \frac{1}{4}L(x-1,y-1) - L(x-1,y+1)} \\ \boldsymbol{- L(x+1,y-1) + L(x+1,y+1)}, \\ \boldsymbol{L_{yy}(x,y) = L(x,y-1) - 2L(x,y) + L(x,y+1)}, \end{aligned} \tag{1.11}$$

що відповідає наступним маскам фільтра:

$$\boldsymbol{L_{xx} = [1 \quad -2 \quad 1]L} \quad \text{та} \quad \boldsymbol{L_{xy} = \begin{bmatrix} -\frac{1}{4} & 0 & \frac{1}{4} \\ 0 & 0 & 0 \\ \frac{1}{4} & 0 & -\frac{1}{4} \end{bmatrix} L} \quad \text{та} \quad \boldsymbol{L_{yy} = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix} L}. \tag{1.12}$$

Похідні вищого порядку для умови знаку третього порядку можна отримати аналогічним чином.

1.5.6 Фазова когерентність

Розробка методів виявлення країв використовує підхід до частотної області для пошуку місцеположень краю. Методи фазової конгруентності

(також відомі як фазова когерентність) намагаються знайти місця на зображенні, де всі синусоїди в частотній області знаходяться у фазі. Ці місця, як правило, будуть відповідати розташуванню сприйманого ребра, незалежно від того, чи ребро представлено великою зміною інтенсивності в просторовій області. Ключовою перевагою цієї техніки є те, що вона сильно реагує на діапазони Маха та уникає помилкових спрацьовувань.

1.5.7 Базування на фізиці

Перетворення фазового розтягування (PST) – це обчислювальний підхід до обробки сигналів та зображень, який базується на фізиці. Одна з його службових програм – виявлення та класифікація ознак. PST є результатом досліджень дисперсійного перетворення Фур'є в часі. PST перетворює зображення шляхом емуляції поширення через дифракційне середовище з розробленою 3D-дисперсійною властивістю (показник заломлення) [8]. Операція покладається на симетрію дисперсійного профілю і може бути зрозуміла з точки зору власних дисперсійних функцій або режимів розтягування. PST виконує подібні функції, як фазово-контрастна мікроскопія, але на цифрових зображеннях. PST також застосовується до цифрових зображень, а також до часових рядів та даних (рис. 1.4).

Рисунок 1.4 – Перетворення за допомогою PST

1.5.8 Субпіксельний підхід

Для підвищення точності розпізнавання контурів було запропоновано декілька субпіксельних методів, включаючи підгонку кривої, моменти, реконструктивні методи та ефекти часткової площі. Ці методи мають різні характеристики. Методи підгонки кривої дуже обчислювальні, але на них легко впливає шум. Методи, засновані на моменті, використовують інтегральний підхід для зменшення ефекту шуму, але в деяких випадках можуть знадобитися додаткові обчислення. Реконструктивні методи використовують горизонтальні градієнти або вертикальні градієнти для побудови кривої та знаходження піку кривої як краю субпікселя. Методи ефекту часткової площі засновані на гіпотезі, згідно з якою значення кожного пікселя залежить від площі по обидва боки ребра всередині цього пікселя, забезпечуючи точну індивідуальну оцінку для кожного пікселя краю. Деякі варіанти техніки, засновані на моменті, виявилися найбільш точними для ізольованих країв [9] (рис. 1.5).

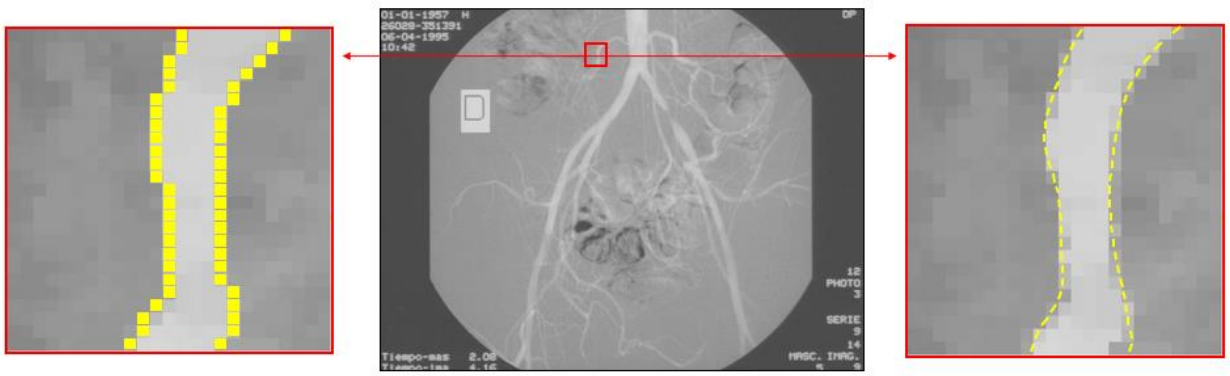

Рисунок 1.5 – Приклад субпіксельного підходу

2 ОГЛЯД ОСНОВНИХ МЕТОДІВ ДЕТЕКТУВАННЯ КОНТУРІВ ОБ'ЄКТІВ

2.1 Техніки ідентифікації контурів

Ідентифікація контурів на зображенні зменшує кількість інформації, яка містить важливі характеристики про форму об'єктів на зображенні.

Контури – це місцеві коливання інтенсивності зображення. Край утворюється між межами двох областей. Основна інформація може бути отримана з краю.

Метод ДК перетворює вихідне зображення в крайове за допомогою операторів. Це добре відомий процес виявлення розривів у значеннях інтенсивності [10].

Процес виявлення країв полягає в перетворюванні зображення в зображення шкали сірого. А потім застосовується детектор контурів для виявлення країв присутніх на зображенні.

Доступні різні методи для виявлення контурів, а саме оператори Робертса, Превітта, Собеля, Щарра, Кірша, Робінсона, Канні, LoG.

2.2 Оператор Робертса

Робертс Крос (оператор Робертса) – один із найдавніших алгоритмів виявлення контурів. Алгоритм був запропонований Лоуренсом Г. Робертсом в 1963 році [11].

Принцип роботи полягає в обчисленні різниці яскравості пікселів, розташованих по діагоналі один біля одного, а потім додавання їх абсолютних значень. Обчислення різниць можна записати як згортку матриці зображення з кожною з матриць 2×2 [12].

Результатом цих операцій є дві матриці, що представляють похідні по напряму, розраховані для напрямків 135° і 45°.

Отримане зображення краю створюється після обчислення відмінностей модуля від відповідних елементів матриці. В оригінальній роботі Робертса представлений інший більш складний метод розрахунку «крутизни» ребра [13]. Обчислення полягає у взятті кореня суми квадратів елементів обох матриць.

Хрест Робертса все ще використовується для швидкості обчислення.

У порівнянні з пізнішими алгоритмами виявлення країв (Превіт, Собель, Шарр), він виявляє нижчу завадостійкість, але дає більш вузькі краї.

Відповідно використовуються наступні маски:

$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}.$$

2.3 Оператор Превітта

Оператор Превітта використовується в цифровій обробці зображень для виявлення вертикальних, горизонтальних і діагональних контурів. Цей оператор дозволяє отримати оцінку похідної по напрямку в одному з 8 напрямків для кожного пікселя зображення [14]. Цей оператор вперше був використаний для аналізу мікроскопічних зображень у медицині Джудіт М. Превіттом в 1966 році.

За простими умовами оператор обчислює градієнт інтенсивності зображення в кожній точці, задаючи напрямок якнайбільшого можливого збільшення від світлого до темного і швидкість зміни в цьому напрямку. Таким чином, результат показує, наскільки «різко» або «плавно» змінюється зображення в цій точці, і, отже, наскільки ймовірно, що частина зображення представляє край, а також як цей край, ймовірно, буде орієнтований. На

практиці обчислення величини (ймовірність ребра) є більш надійним і легшим для інтерпретації, ніж розрахунок напрямку.

Математично градієнт функції із двома змінними (тут функція інтенсивності зображення) є у кожній точці зображення двовимірним вектором із компонентами, заданими похідними в горизонтальному та вертикальному напрямках. У кожній точці зображення вектор градієнта вказує у напрямку найбільшого можливого збільшення інтенсивності, а довжина вектора градієнта відповідає швидкості зміни в цьому напрямку. Це означає, що результат оператора Превітта в точці зображення, яка знаходиться в області постійної інтенсивності зображення, є нульовим вектором, а в точці на ребрі – вектор, який вказує через край, від темних до яскравих значень.

Відповідно використовуються наступні маски:

$$\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \quad \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}.$$

2.4 Оператор Собеля

Оператор Собеля – один з операторів, який використовується для ідентифікації контурів при цифровій обробці зображень. Це дискретний похідний оператор, який дозволяє наблизити похідні від напрямків інтенсивності зображення у восьми напрямках кожні 45°. Вперше цей оператор був представлений у статті Ірвіна Собеля (співавтор Джером Артур Фельдман), текст якої не публікувався, проте, широко цитується [15].

Оператор Собеля обчислюється повільніше, ніж оператор Робертса Кроса, але його більше ядро згортки більшою мірою згладжує вхідне зображення, що робить оператор менш чутливим до шуму. Оператор також,

як правило, виробляє значно більші вихідні значення для подібних ребер, порівняно з оператором Робертса.

Як і у випадку з оператором Робертса, вихідні значення оператора можуть легко перевищити максимально дозволене значення пікселів для типів зображень, які підтримують лише незначні цілочисельні значення пікселів (наприклад, 8-бітові цілочисельні зображення) [16]. Коли це трапляється, стандартною практикою є просто встановити переповнені вихідні пікселі на максимально допустиме значення. Проблеми можна уникнути, використовуючи тип зображення, що підтримує значення пікселів із більшим діапазоном.

Природні краї зображень часто ведуть до ліній вихідного зображення, які мають ширину в кілька пікселів завдяки ефекту згладжування оператора Собеля. Для протидії цьому може бути бажаним деяке стоншення. Якщо цього не вдасться, можна буде використати якесь відстеження гістерезисних хребтів, як у оператора Кенні.

Відповідно використовуються наступні маски:

$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}.$$

2.5 Оператор Шарра

Один з операторів, який використовується для ІК при цифровій обробці зображень. Це дискретний похідний оператор, який дозволяє наблизити похідні від напрямків інтенсивності зображення у восьми напрямках кожні 45°. Цей оператор був представлений Ганно Шарром у своїй докторській дисертації у 2000 році [17].

Оператор Шарра є результатом оптимізації, що мінімізує зважену середньоквадратичну кутову помилку в області Фур'є. Ця оптимізація проводиться за умови, що отримані фільтри є чисельно узгодженими.

Тому вони насправді є похідними ядрами, а не просто дотримуються обмежень симетрії.

Подібну стратегію оптимізації та отримані фільтри також представили Фарід та Сімончеллі. Вони також досліджують схеми похідних вищого порядку. На відміну від оператор Шарра, ці фільтри не застосовуються, щоб бути чисельно узгодженими [18].

Проблема конструкції похідного фільтра була переглянута, наприклад від Круна.

Похідні фільтри на основі довільних кубічних сплайнів були представлені Хастом. Він показав, як похідні першого та другого порядку можуть бути правильно обчислені за допомогою кубічних або тригонометричних сплайнів за допомогою методу подвійної фільтрації, що дає фільтри довжиною 7.

Оптимальні для орієнтації похідні ядра різко зменшують систематичні похибки оцінки при оцінці оптичного потоку. Більші схеми з ще більшою точністю та оптимізованими сімействами фільтрів для розширеної оцінки оптичного потоку були представлені в подальшій роботі Шарра [19]. Набори похідних фільтрів другого порядку досліджені для оцінки прозорого руху. Було помічено, що чим більші отримані ядра, тим краще вони наближають похідну від гауссових фільтрів.

Відповідно використовуються наступні маски:

$$\begin{bmatrix} 3 & 0 & -3 \\ 10 & 0 & -10 \\ 3 & 0 & -3 \end{bmatrix} \quad \begin{bmatrix} 3 & 10 & 3 \\ 0 & 0 & 0 \\ -3 & -10 & -3 \end{bmatrix}.$$

2.6 Оператор Кірша

Оператор, запропонований Кіршем, знаходить максимальний відгук краю у кількох заздалегідь визначених напрямках. Оператор має одну ядро і обертається з кроком в 45° через усі 8 напрямків компаса, тобто *N, NW, W, SW, S, SE, E, NE*. Крайня величина обчислюється як максимальна величина за всіма напрямками. Усі пікселі зображень використовують маски, щоб зробити згортку, кожна маска має відповідь на певний напрямок краю, максимальне значення для всіх 8 напрямків встановлює вихідне значення цієї точки.

Алгоритм Кірша базується на кроці, відповідно до характеристик самих зображень та умов оцінки значень Кірша, він може регулювати порогові значення щоб отримати максимально можливу точку краю зображень, щоб він міг повністю відірватися від штучної участі [20].

Алгоритм Кірша може регулювати порогові значення, щоб отримати максимально можливу точку краю зображення, тому його можна повністю відірвати від штучної участі. Змінні, що стосуються вибору оператора виявлення ребра, складаються з орієнтації краю, структури ребра та шумового середовища.

Тому оператори можуть бути оптимізовані для пошуку вертикальних, горизонтальних та діагональних ребер [21]. Прийоми, що застосовуються на шумних зображення, як правило, більші за обсягом, і алгоритм може бути сприйнятливим до такої регулярної зміни в таких випадках.

Відповідно використовується наступна маска яку обертають по 8 напрямкам:

$$\begin{bmatrix} 5 & 5 & 5 \\ -3 & 0 & -3 \\ -3 & -3 & -3 \end{bmatrix}.$$

2.7 Оператор Робінсона.

Маски компаса Робінсона – це ще один тип маски, який використовується для виявлення країв. Цей оператор також відомий як маска напрямку [22]. У цьому операторі необхідно взяти одну маску і обертати її у всіх 8 основних напрямках компаса.

Оператор не має фіксованої маски. Можна взяти будь-яку маску, і далі потрібно повернути її, щоб знайти краї у всіх вищезазначених напрямках. Всі маски обертаються на основах напрямку нульових стовпців.

Вісь напрямку – це лінія нулів у матриці. Маска компаса Робінсона схожа на маски компаса Кірша, але є простішою у застосуванні. Оскільки матричні коефіцієнти містять лише 0, 1, 2 і є симетричними, потрібно обчислювати лише результати чотирьох масок, інші чотири результати є запереченням перших чотирьох результатів. Край або контур – це крихітна область із сусідніми різними значеннями пікселів [23]. Згортання кожної маски із зображенням створює вихід великих значень там, де відбувається швидка зміна значення пікселя, таким чином знайдено крайову точку. Всі виявлені крайові точки будуть вибудовуватися як ребра.

Відповідно використовується наступна маска яку обертають по 8 напрямкам:

$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}.$$

2.8 Постановка задачі дослідження

Таким чином, знаходження геометричних примітивів є актуальним завданням для обробки і розпізнавання зображень. Тому ставиться завдання дослідження та реалізації алгоритмів ідентифікації контурів, з використанням операторів Робертса, Превітта, Собеля, Шарра, Кірша та Робінсона.

В рамках роботи об’єктом дослідження виступають алгоритми (оператори) за допомогою яких здійснюється розпізнавання образів на зображенні та подальше використання отриманої інформації для аналізу об’єктів.

Метою дослідження є аналіз та порівняння існуючих операторів в критеріях швидкості та складності реалізації, а також модифікація обраних операторів за допомогою видалення або придушення шумів для більш якісного результату.

Для цього необхідно вирішити такі завдання:

- провести аналіз існуючих алгоритмів ІК об’єктів на зображенні;
- реалізувати та модифікувати алгоритми ІК;
- розробити комп’ютерну модель для ІК об’єктів на зображенні.

3 МАТЕМАТИЧНІ МОДЕЛІ ОПЕРАТОРІВ ДЕТЕКТУВАННЯ КОНТУРІВ ОБ'ЄКТІВ

3.1 Оператор Робертса

На думку Робертса, детектор країв повинен мати такі властивості: створювані краї повинні бути чітко визначеними, фон повинен забезпечувати якомога менше шуму, а інтенсивність країв повинна відповідати якомога ближче до того, що сприймає людина. З урахуванням цих критеріїв і спираючись на переважаючу тоді психофізичну теорію, Робертс запропонував такі рівняння [24]:

$$\boldsymbol{y_{i,y}=\sqrt{x_{i,j}}}, \tag{3.1}$$

де *x* – початкове значення інтенсивності на зображенні;

z – обчислена похідна;

i, j – місце розташування на зображенні.

Результати цієї операції дозволять виділити зміни інтенсивності в діагональному напрямку. Одним з найбільш привабливих аспектів цієї операції є її простота; ядро невелике і містить лише цілі числа. Однак із швидкістю роботи комп'ютерів сьогодні ця перевага незначна, і оператор Робертса сильно страждає від чутливості до шуму [25].

Для того, щоб виконати ДК за допомогою оператора Робертса, необхідно спочатку конвертувати вихідне зображення за наступними двома ядрами:

$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}.$$

Нехай $I(x,y)$ – точка на вихідному зображенні, а $G_x(x,y)$ – точка на зображенні, сформованому згортанням з першим ядром, а $G_y(x,y)$ – точка на

зображенні, сформована згортанням з другим ядром. Потім градієнт можна визначити як:

$$\boldsymbol{\nabla I(x,y) = G(x,y) = \sqrt{G_x^2 + G_y^2}}. \tag{3.2}$$

Напрямок градієнта також можна визначити наступним чином:

$$\boldsymbol{\theta(x,y) = \arctan\left(\frac{G_y(x,y)}{G_x(x,y)}\right) - \frac{3\pi}{4}}. \tag{3.3}$$

Зверніть увагу, що кут 0° відповідає вертикальній орієнтації, така що напрямок максимального контрасту від чорного до білого проходить зліва направо на зображенні.

3.2 Оператор Превітта

Простіше кажучи, оператор обчислює градієнт інтенсивності зображення в кожній точці, надаючи напрямок максимально можливого збільшення від світлого до темного і швидкість зміни в цьому напрямку. Таким чином, результат показує, наскільки «різко» або «плавно» змінюється зображення в цій точці, і, отже, наскільки ймовірно, що частина зображення представляє край, а також як цей край, ймовірно, буде орієнтований. На практиці обчислення величини (ймовірність ребра) є більш надійним і легшим для інтерпретації, ніж розрахунок напрямку [26].

Математично градієнт функції із двома змінними (тут функція інтенсивності зображення) є у кожній точці зображення двовимірним вектором із компонентами, заданими похідними в горизонтальному та вертикальному напрямках. У кожній точці зображення вектор градієнта

вказує у напрямку найбільшого можливого збільшення інтенсивності, а довжина вектора градієнта відповідає швидкості зміни в цьому напрямку. Це означає, що результат оператора Превітта в точці зображення, яка знаходиться в області постійної інтенсивності зображення, є нульовим вектором, а в точці на ребрі – вектор, який вказує через край, від темних до яскравих значень.

Математично, оператор використовує два ядра 3×3, для обчислення наближень похідних – одне для горизонтальних змін, а інше для вертикальних. Якщо визначити A як вихідне зображення, і G_x та G_y – це два зображення, які в кожній точці містять горизонтальне та вертикальне наближення, похідні обчислюються як [27]:

$$\boldsymbol{G_x} = \begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{-1} \\ \mathbf{1} & \mathbf{0} & \mathbf{-1} \\ \mathbf{1} & \mathbf{0} & \mathbf{-1} \end{bmatrix} * \boldsymbol{A}, \qquad \boldsymbol{G_y} = \begin{bmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{-1} & \mathbf{-1} & \mathbf{-1} \end{bmatrix} * \boldsymbol{A}, \tag{3.4}$$

де операція «*» тут означає двовимірну операцію згортки.

Оскільки ядра Превітта можна розкласти як добутки ядра усереднення та диференціації, вони обчислюють градієнт із згладжуванням. Отже, це відокремлюваний фільтр. Наприклад, G_x можна записати як:

$$\begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{-1} \\ \mathbf{1} & \mathbf{0} & \mathbf{-1} \\ \mathbf{1} & \mathbf{0} & \mathbf{-1} \end{bmatrix} = \begin{bmatrix} \mathbf{1} \\ \mathbf{1} \\ \mathbf{1} \end{bmatrix} [\mathbf{1} \quad \mathbf{0} \quad \mathbf{-1}], \tag{3.5}$$

де x координата тут визначається як зростаюча в «лівому» напрямку;

y координата визначається як зростаюча в «верхньому» напрямку.

У кожній точці зображення отримані апроксимації градієнта можна об’єднати, щоб отримати величину градієнта, використовуючи:

$$\boldsymbol{G} = \sqrt{\boldsymbol{G_x^2} + \boldsymbol{G_y^2}}. \tag{3.6}$$

Використовуючи цю інформацію, можна також розрахувати напрямок градієнта:

$$\boldsymbol{\theta} = \mathbf{atan2}\left(\boldsymbol{G_y}, \boldsymbol{G_x}\right), \tag{3.7}$$

де Θ дорівнює 0 для вертикального краю, який темніший з правого боку.

3.3 Оператор Собеля

Оператор використовує два ядра 3×3, які скручені з вихідним зображенням, для обчислення наближень похідних – одне для горизонтальних змін, а одне для вертикальних. Якщо визначити *A* як вихідне зображення, а G_x і G_y – це два зображення, які в кожній точці містять вертикальне та горизонтальне наближення похідних відповідно, обчислення наступні:

$$\boldsymbol{G_x} = \begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{-1} \\ \mathbf{2} & \mathbf{0} & \mathbf{-2} \\ \mathbf{1} & \mathbf{0} & \mathbf{-1} \end{bmatrix} * \boldsymbol{A}, \qquad \boldsymbol{G_y} = \begin{bmatrix} \mathbf{1} & \mathbf{2} & \mathbf{1} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{-1} & \mathbf{-2} & \mathbf{-1} \end{bmatrix} * \boldsymbol{A}, \tag{3.8}$$

де операція «*» тут позначає двовимірну операцію згортки обробки сигналу.

Оскільки ядра Собеля можна розкласти як добутки ядра усереднення та диференціації, вони обчислюють градієнт із згладжуванням. Наприклад, G_x можна записати як [28]:

$$\boldsymbol{G_x} = \begin{bmatrix} \mathbf{1} \\ \mathbf{2} \\ \mathbf{1} \end{bmatrix} * ([\mathbf{1} \quad \mathbf{0} \quad -\mathbf{1}] * \boldsymbol{A}), \qquad \boldsymbol{G_y} = \begin{bmatrix} \mathbf{1} \\ \mathbf{0} \\ -\mathbf{1} \end{bmatrix} * ([\mathbf{1} \quad \mathbf{2} \quad \mathbf{1}] * \boldsymbol{A}), \tag{3.9}$$

де *x* координата тут визначається як зростаюча в «правому» напрямку;

а *y* координата визначається як зростаюча в «верхньому» напрямку.

У кожній точці зображення отримані апроксимації градієнта можна об’єднати, щоб отримати величину градієнта, використовуючи:

$$\boldsymbol{G} = \sqrt{\boldsymbol{G_x^2} + \boldsymbol{G_y^2}}. \tag{3.10}$$

Використовуючи цю інформацію, також можна розрахувати напрямок градієнта:

$$\boldsymbol{\theta} = \mathbf{atan}\left(\frac{\boldsymbol{G_y}}{\boldsymbol{G_x}}\right), \tag{3.11}$$

де Θ дорівнює 0 для вертикального краю, який темніший з правого боку.

Оскільки функція інтенсивності цифрового зображення відома лише в дискретних точках, похідні від цієї функції неможливо визначити, якщо не припустити, що існує основна диференційована функція інтенсивності, яка була відібрана в точках зображення. З деякими додатковими припущеннями, похідну від функції безперервної інтенсивності можна обчислити як функцію від вибіркової функції інтенсивності, тобто цифрового зображення. Виявляється, що похідні в будь-якій конкретній точці є функціями значень інтенсивності практично у всіх точках зображення. Однак наближення цих похідних функцій можна визначити з меншим або більшим ступенем точності [29].

Оператор Собеля-Фельдмана представляє досить неточне наближення градієнта зображення, але все ще є достатньо якісним, щоб мати практичну користь у багатьох додатках [30]. Точніше, він використовує значення інтенсивності лише в області 3×3 навколо кожної точки зображення, щоб наблизити відповідний градієнт зображення, і використовує лише цілі значення для коефіцієнтів, які зважують інтенсивність зображення, для отримання апроксимації градієнта.

Оператор Собеля-Фельдмана складається з двох відокремлюваних операцій:

– згладжування перпендикулярно напрямку похідної за допомогою трикутного фільтра: $h(-1) = 1,\ h(0) = 2,\ h(1) = 1$;

– проста центральна різниця у напрямку похідної: $h'(-1) = 1,\ h'(0) = 0,\ h'(1) = -1$.

Фільтри Собеля-Фельдмана для похідних зображень різних розмірів із $x, y, z, t \in \{-1, 0, 1\}$:

– 1D: $h'_x(x) = h'(x)$;

– 2D: $h'_x(x, y) = h'(x)h(y)$;

– 3D: $h'_x(x, y, z) = h'(x)h(y)h(z)$;

– 4D: $h'_x(x, y, z, t) = h'(x)h(y)h(z)$.

Отже, як приклад ядро 3D Собеля-Фельдмана в напрямі z дорівнює:

$$h'_z(:,:,-1) = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}, \qquad h'_z(:,:,0) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

$$h'_z(:,:,1) = \begin{bmatrix} -1 & -2 & -1 \\ -2 & -4 & -2 \\ -1 & -2 & -1 \end{bmatrix}.$$

Як наслідок його визначення, оператор Собеля може бути реалізований простими засобами як в апаратному, так і в програмному забезпеченні: для обчислення відповідного результату потрібно лише вісім точок зображення

навколо точки, а для обчислення наближення вектора градієнта потрібна лише ціла арифметика [31].

Крім того, обидва описані вище дискретні фільтри можна розділити:

$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} [1 \quad 0 \quad -1] = \begin{bmatrix} 1 \\ 1 \end{bmatrix} * \begin{bmatrix} 1 \\ 1 \end{bmatrix} [1 \quad -1] * [1 \quad 1],$$

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} [1 \quad 2 \quad 1] = \begin{bmatrix} 1 \\ 1 \end{bmatrix} * \begin{bmatrix} 1 \\ -1 \end{bmatrix} [1 \quad 1] * [1 \quad 1].$$

і дві похідні G_x і G_y тому можна обчислити як:

$$\begin{aligned} \boldsymbol{G_x} &= \begin{bmatrix} \mathbf{1} \\ \mathbf{2} \\ \mathbf{1} \end{bmatrix} * ([\mathbf{1} \quad \mathbf{0} \quad -\mathbf{1}] * \boldsymbol{A}), \qquad \boldsymbol{G_y} \\ &= \begin{bmatrix} \mathbf{1} \\ \mathbf{0} \\ -\mathbf{1} \end{bmatrix} * ([\mathbf{1} \quad \mathbf{2} \quad \mathbf{1}] * \boldsymbol{A}). \end{aligned} \tag{3.12}$$

У певних реалізаціях це роздільне обчислення може бути вигідним, оскільки воно передбачає менше арифметичних обчислень для кожної точки зображення.

3.4 Оператор Шарра

Оператор використовує два ядра 3×3, які скручені з вихідним зображенням, для обчислення наближень похідних – одне для горизонтальних змін, а одне для вертикальних.

Якщо визначити A як вихідне зображення, а G_x і G_y – це два зображення, які в кожній точці містять вертикальне та горизонтальне наближення похідних відповідно, обчислення наступні [32]:

$$\boldsymbol{G_x} = \begin{bmatrix} \mathbf{3} & \mathbf{0} & \mathbf{-3} \\ \mathbf{10} & \mathbf{0} & \mathbf{-10} \\ \mathbf{3} & \mathbf{0} & \mathbf{-3} \end{bmatrix} * \boldsymbol{A}, \qquad \boldsymbol{G_y} = \begin{bmatrix} \mathbf{3} & \mathbf{10} & \mathbf{3} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{-3} & \mathbf{-10} & \mathbf{-3} \end{bmatrix} * \boldsymbol{A}, \tag{3.13}$$

де операція «*» тут позначає двовимірну операцію згортки обробки сигналу.

Оскільки ядра Собеля можна розкласти як добутки ядра усереднення та диференціації, вони обчислюють градієнт із згладжуванням. Наприклад, G_x можна записати як [33]:

$$\begin{aligned} \boldsymbol{G_x} &= \begin{bmatrix} \mathbf{3} \\ \mathbf{10} \\ \mathbf{3} \end{bmatrix} * ([\mathbf{3} \quad \mathbf{0} \quad \mathbf{-3}] * \boldsymbol{A}), \\ \boldsymbol{G_y} &= \begin{bmatrix} \mathbf{3} \\ \mathbf{0} \\ \mathbf{-3} \end{bmatrix} * ([\mathbf{3} \quad \mathbf{10} \quad \mathbf{3}] * \boldsymbol{A}), \end{aligned} \tag{3.14}$$

де *x* координата тут визначається як зростаюча в «правому» напрямку;

y координата визначається як зростаюча в «зростаючому» напрямку.

У кожній точці зображення отримані апроксимації градієнта можна об’єднати, щоб отримати величину градієнта, використовуючи:

$$\boldsymbol{G} = \sqrt{\boldsymbol{G_x^2} + \boldsymbol{G_y^2}}. \tag{3.15}$$

Використовуючи цю інформацію, також можна розрахувати напрямок градієнта:

$$\boldsymbol{\theta} = \mathbf{atan}\left(\frac{\boldsymbol{G_y}}{\boldsymbol{G_x}}\right), \tag{3.16}$$

де Θ дорівнює 0 для вертикального краю, який темніший з правого боку.

Оператор Собеля-Фельдмана, хоча і зменшує артефакти, пов’язані з оператором суто центральних відмінностей, але він не має ідеальної обертової симетрії.

Оператор Шарра є результатом оптимізації, що мінімізує зважену середньоквадратичну кутову помилку в області Фур’є. Ця оптимізація проводиться за умови, що отримані фільтри є чисельно узгодженими. Тому вони насправді є похідними ядрами, а не просто дотримуються обмежень симетрії. Оптимальним 8-бітовим цілочисельним фільтром 3×3, що випливає з теорії Шарра, є [34]:

$$\begin{bmatrix} 47 & 0 & -47 \\ 162 & 0 & -162 \\ 47 & 0 & -47 \end{bmatrix} \quad \begin{bmatrix} 47 & 162 & 47 \\ 0 & 0 & 0 \\ -47 & -162 & -47 \end{bmatrix}.$$

Подібну стратегію оптимізації та отримані фільтри також представили Фарід та Сімончеллі. Вони також досліджують схеми похідних вищого порядку. На відміну від роботи Шарра, ці фільтри не застосовуються, щоб бути чисельно узгодженими.

3.5 Оператор Кірша

Оператор бере одиночну маску ядра і обертає її з кроком у 45° за всіма 8 напрямками компаса: *N*, *NW*, *W*, *SW*, *S*, *SE*, *E*, *NE*. Крайова величина оператора Кірша обчислюється як максимальна величина в усіх напрямках:

$$\boldsymbol{h_{n,m} = \max_{z=1,\dots,8} \sum_{i=-1}^{1} \sum_{j=-1}^{1} g_{ij}^{(z)} * f_{n+i,m+j},} \tag{3.17}$$

де z перераховує напрямки компаса ядра g:

$$g^{(1)} = \begin{bmatrix} 5 & 5 & 5 \\ -3 & 0 & -3 \\ -3 & -3 & -3 \end{bmatrix}, \quad g^{(2)} = \begin{bmatrix} 5 & 5 & -3 \\ 5 & 0 & -3 \\ -3 & -3 & -3 \end{bmatrix}, \quad g^{(3)} = \begin{bmatrix} 5 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & -3 & -3 \end{bmatrix},$$

$$g^{(4)} = \begin{bmatrix} -3 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & 5 & -3 \end{bmatrix}, \quad g^{(5)} = \begin{bmatrix} -3 & -3 & -3 \\ -3 & 0 & -3 \\ 5 & 5 & 5 \end{bmatrix}, \quad g^{(6)} = \begin{bmatrix} -3 & -3 & -3 \\ -3 & 0 & 5 \\ -3 & 5 & 5 \end{bmatrix},$$

$$g^{(7)} = \begin{bmatrix} -3 & -3 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & 5 \end{bmatrix}, \quad g^{(8)} = \begin{bmatrix} -3 & 5 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & -3 \end{bmatrix}.$$

Напрямок краю визначається маскою, яка створює максимальну величину краю.

3.6 Оператор Робінсона

Оператор бере одиночну маску ядра і обертає її з кроком у 45° за всіма 8 напрямками компаса: *N*, *NW*, *W*, *SW*, *S*, *SE*, *E*, *NE*. Крайова величина оператора Кірша обчислюється як максимальна величина в усіх напрямках:

$$\boldsymbol{h_{n,m} = \max_{z=1,\dots,8} \sum_{i=-1}^{1} \sum_{j=-1}^{1} g_{ij}^{(z)} * f_{n+i,m+j},} \tag{3.18}$$

де z перераховує напрямки компаса ядра g [10]:

$$g^{(1)} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad g^{(2)} = \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix}, \quad g^{(3)} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix},$$

$$g^{(4)} = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -2 \end{bmatrix}, \quad g^{(5)} = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}, \quad g^{(6)} = \begin{bmatrix} 0 & -1 & -2 \\ 1 & 0 & -1 \\ 2 & 1 & 0 \end{bmatrix},$$

$$g^{(7)} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}, \quad g^{(8)} = \begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ - & 1 & 2 \end{bmatrix}.$$

Напрямок краю визначається маскою, яка створює максимальну величину краю.

Реалізація оператора Робінсона враховує можливість його застосування спільно з оператором Маслова [35]. Зокрема, оператор Робінсона може застосовуватися не тільки по контрарній парі, як у класичному методі, але і з будь-яким сприятливим набором, який містить один предикат або більш ніж два предиката.

4 КОМП'ЮТЕРНА МОДЕЛЬ ДЕТЕКТУВАННЯ КОНТУРІВ

4.1 Обґрунтування вибору середовища програмної реалізації

У рамках атестаційної роботи було розроблено алгоритми для ДК за допомогою операторів ідентифікації контурів об'єктів на зображенні. Для реалізації було обране середовище IDE WebStorm, мову програмування JavaScript та допоміжні бібліотеки або фреймворки PIXI.js та React.js.

4.1.1 IDE WebStorm

WebStorm – це інтегроване середовище розробки на JavaScript, CSS & HTML від компанії JetBrains, розроблена на основі платформи IntelliJ IDEA.

Він забезпечує автодоповнення, аналіз коду на льоту, навігацію по коду, рефакторинг, налагодження, і інтеграцію з системами управління версіями. Важливою перевагою інтегрованого середовища розробки WebStorm є робота з проектами (в тому числі, рефакторинг коду JavaScript, що знаходиться в різних файлах і папках проекту, а також вкладеного в HTML). Підтримується множинна вкладеність (коли в документ на HTML вкладений скрипт на Javascript, в який вкладено інший код HTML, всередині якого вкладено Javascript) – тобто в таких конструкціях підтримується коректний рефакторинг.

4.1.2 JavaScript

В якості мови програмування було обрано МП JavaScript. Це мультіпарадігменна мова програмування. Підтримує об'єктно-орієнтований,

імперативний і функціональний стилі. Є реалізацією стандарту ECMAScript (стандарт ECMA-262).

JavaScript зазвичай використовується як вбудована мова для програмного доступу до об’єктів додатків. Найбільш широке застосування знаходить в браузерах як мова сценаріїв для додання інтерактивності веб-сторінкам.

Основні архітектурні риси: динамічна типізація, слабка типізація, автоматичне керування пам’яттю, прототипне програмування, функції як об’єкти першого класу.

На JS вплинули багато мов, при розробці була мета зробити мову схожим на Java. Мовою JavaScript не володіє будь-яка компанія або організація, що відрізняє його від ряду мов програмування, використовуваних в веб-розробці.

JavaScript є об’єктно-орієнтованою мовою, але що використовується в мові обумовлює відмінності в роботі з об’єктами в порівнянні з традиційними клас-орієнтованими мовами. Крім того, JS має ряд властивостей, властивих функціональним мовам, – функції як об’єкти першого класу, об’єкти як списки, каррінг, анонімні функції, замикання – що додає мові додаткову гнучкість.

Незважаючи на схожий з C синтаксис, JavaScript в порівнянні з мовою C має корінні відмінності:

– об’єкти з можливістю інтроспекції;
– функції як об’єкти першого класу;
– автоматичне приведення типів;
– автоматичне прибирання сміття;
– анонімні функції.

У мові відсутні такі корисні речі, як:

– стандартна бібліотека: зокрема, відсутній інтерфейс програмування додатків по роботі з файловою системою, управління потоками введення-виведення, базових типів для бінарних даних;

– стандартні інтерфейси до веб-серверів і баз даних;

– система управління пакетами, яка б відстежувала залежності і автоматично встановлювала їх.

4.1.3 PIXI.js

PIXI.js – це бібліотека, яка реалізує швидкий і простий механізм візуалізації. Вона може працювати в поєднанні з рядом інших бібліотек на JavaScript і виконувати візуалізацію на основі canvas і WebGL.

Це неймовірно швидкий двигун рендерінгу 2D графічних об’єктів. Він дозволяє відображати, анімувати і управляти інтерактивною графікою, легко створювати ігри та програми з використанням JS та інших технологій HTML5. Він має хороший API і включає в себе безліч корисних функцій, таких як підтримка текстурних атласів і надання оптимізованої системи для анімації спрайтів (інтерактивних зображень). Він також дає вам повні граф сцени, так що можна створювати ієрархії вбудованої графіки (спрайтів всередині спрайтів), а також дозволяє прикріплювати події миші і торкання безпосередньо до спрайтів.

4.1.4 React.js

JavaScript-бібліотека з відкритим вихідним кодом для розробки призначених для користувача інтерфейсів.

React розробляється і підтримується Facebook, Instagram і співтовариством окремих розробників і корпорацій.

React може використовуватися для розробки односторінкових і мобільних додатків. Його мета – надати високу швидкість, простоту і масштабованість. Як бібліотека для розробки призначених для користувача

інтерфейсів React часто використовується з іншими бібліотеками, такими як MobX, Redux і GraphQL.

4.2 Програмна реалізація

Було розглянуто декілька алгоритмів просторової фільтрації для аналізу контуру зображення. Кожен з них відрізняється загалом маскою (або ядром, фільтром) яка застосовується для вихідного зображення.

В даній атестаційній роботі було обрано оператори Роберста, Превітта, Собеля, Шарра, Кірша та Робінсона які дозволяють знайти наближені значення градієнта яскравості зображення, та проаналізувати зображення на наявність контурів об’єктів.

Більшість алгоритмів розпізнавання образів вимагають попереднього перетворення вхідного зображення. Деяким досить простого Grayscale перетворення [36].

Grayscale – це одне з часто вживаних перетворень, за допомогою якого з кольорового зображення можна отримати зображення в відтінках сірого [37]. Дане перетворення сприяє зменшенню кількості даних на зображенні, нам не доводиться враховувати колір джерел світла які впливають на об’єкти.

$$\boldsymbol{Gray} = \frac{\boldsymbol{R} + \boldsymbol{G} + \boldsymbol{B}}{\boldsymbol{3}}. \tag{4.1}$$

Програмна реалізація даного етапу (рис. 4.1).

```
const getAveragePixelColor = (x: number, y: number): number => {
  const index = (y * outputImage.width + x) * 4;
  return (data[index] + data[index + 1] + data[index + 2]) / 3;
};
```

Рисунок 4.1 – Перетворення кольорового пікселю в значення сірого градієнту

В якості модифікації алгоритмів ДК було обрано медіанний фільтр для зменшення рівня шуму на зображенні.

Медіанний фільтр – один з видів цифрових фільтрів, широко використовуваний в цифровій обробці сигналів і зображень для зменшення рівня шуму. Медіанний фільтр є нелінійним КИХ-фільтром [38].

Значення відліків усередині вікна фільтра сортуються в порядку зростання (спадання); і значення, що знаходиться в середині упорядкованого списку, надходить на вихід фільтра. У разі парного числа відліків у вікні вихідне значення фільтра дорівнює середньому значенню двох відліків в середині упорядкованого списку. Вікно переміщається уздовж фільтруючого сигналу і обчислення повторюються.

Медіанна фільтрація – ефективна процедура обробки сигналів, що піддаються впливу імпульсних перешкод (рис. 4.2) [39].

```
const windowSize = 5;
const window: number[] = new Array( arrayLength: windowSize * windowSize);
const edgeX = Math.floor( x: windowSize / 2);
const edgeY = Math.floor( x: windowSize / 2);
for (let x = edgeX; x < image.width - edgeX; x++) {
  for (let y = edgeY; y < image.height - edgeY; y++) {
    let i = 0;
    for (let fx = 0; fx < windowSize; fx++) {
      for (let fy = 0; fy < windowSize; fy++) {
        window[i++] = getAveragePixelColor( x: x + fx - edgeX,  y: y + fy - edgeY);
      }
    }
    window.sort( compareFn: (a : number , b : number ) => a - b);
    setPixelColor(x, y, window[Math.floor( x: windowSize * windowSize / 2)]);
  }
}
```

Рисунок 4.2 – Програмна реалізація медіанного фільтру

Як тільки було зменшено кількість шуму, необхідно застосувати один із алгоритмів ІК. В ході роботи було обрано 6 (шість) операторів для отримання необхідного результату.

Оператори Робертса, Превітта, Собеля та Шарра використовують маску (ядро) 3×3. В програмному вигляді маємо наступне (рис. 4.3).

```
for (let x = 0; x < image.width; x++) {
  for (let y = 0; y < image.height; y++) {
    let Gx, Gy;
    if (x <= 1 || y <= 1 || x >= image.width - 2 || y >= image.height - 2) {
      Gx = 0;
      Gy = 0;
    } else {
      let f00 = image.getPixelColor(x, y);
      let f01 = image.getPixelColor(x,  y: y + 1);
      let f02 = image.getPixelColor(x,  y: y + 2);

      let f10 = image.getPixelColor( x: x + 1, y);
      let f11 = image.getPixelColor( x: x + 1,  y: y + 1);
      let f12 = image.getPixelColor( x: x + 1,  y: y + 2);

      let f20 = image.getPixelColor( x: x + 2, y);
      let f21 = image.getPixelColor( x: x + 2,  y: y + 1);
      let f22 = image.getPixelColor( x: x + 2,  y: y + 2);

      Gx =
        (X[0][0] * f00) + (X[0][1] * f10) + (X[0][2] * f20) +
        (X[1][0] * f01) + (X[1][1] * f11) + (X[1][2] * f21) +
        (X[2][0] * f02) + (X[2][1] * f12) + (X[2][2] * f22);

      Gy =
        (Y[0][0] * f00) + (Y[0][1] * f10) + (Y[0][2] * f20) +
        (Y[1][0] * f01) + (Y[1][1] * f11) + (Y[1][2] * f21) +
        (Y[2][0] * f02) + (Y[2][1] * f12) + (Y[2][2] * f22);
    }

    let G = Math.abs(Gx) + Math.abs(Gy);

    if (G > 255)
      G = 255;
```

Рисунок 4.3 – Програмна реалізація операторів Робертса, Превітта, Собеля та Шарра

Оператори Кірша та Робінсоса використовують маску (ядро) з розмірністю 3×3, але їх крутять по 8 (восьми) напрямкам компасу: *N*, *NW*, *W*, *SW*, *S*, *SE*, *E*, *NE* (рис. 4.4) [40].

```
const f = (mask: any,
           f00: number, f01: number, f02: number,
           f10: number, f11: number, f12: number,
           f20: number, f21: number, f22: number) => {
  return (mask[0][0] * f00) + (mask[0][1] * f10) + (mask[0][2] * f20) +
    (mask[1][0] * f01) + (mask[1][1] * f11) + (mask[1][2] * f21) +
    (mask[2][0] * f02) + (mask[2][1] * f12) + (mask[2][2] * f22);
};

for (let x = 0; x < image.width; x++) {
  for (let y = 0; y < image.height; y++) {
    const gradients = [];
    if (x <= 1 || y <= 1 || x >= image.width - 2 || y >= image.height - 2) {
    } else {
      let f00 = image.getPixelColor(x, y);
      let f10 = image.getPixelColor( x: x + 1, y);
      let f20 = image.getPixelColor( x: x + 2, y);

      let f01 = image.getPixelColor(x,  y: y + 1);
      let f11 = image.getPixelColor( x: x + 1,  y: y + 1);
      let f21 = image.getPixelColor( x: x + 2,  y: y + 1);

      let f02 = image.getPixelColor(x,  y: y + 2);
      let f12 = image.getPixelColor( x: x + 1,  y: y + 2);
      let f22 = image.getPixelColor( x: x + 2,  y: y + 2);

      gradients.push(f(E, f00, f01, f02, f10, f11, f12, f20, f21, f22));
      gradients.push(f(NE, f00, f01, f02, f10, f11, f12, f20, f21, f22));
      gradients.push(f(N, f00, f01, f02, f10, f11, f12, f20, f21, f22));
      gradients.push(f(NW, f00, f01, f02, f10, f11, f12, f20, f21, f22));
      gradients.push(f(W, f00, f01, f02, f10, f11, f12, f20, f21, f22));
      gradients.push(f(SW, f00, f01, f02, f10, f11, f12, f20, f21, f22));
      gradients.push(f(S, f00, f01, f02, f10, f11, f12, f20, f21, f22));
      gradients.push(f(SE, f00, f01, f02, f10, f11, f12, f20, f21, f22));
    }

    let G = _.max(gradients)!;

    if (G > 255)
      G = 255;
```

Рисунок 4.4 – Програмна реалізація операторів Кірша та Робінсона

4.3 Інструкція користувача

Програма має наступний вигляд (рис. 4.5).

Рисунок 4.5 – Початковий вигляд програми

Де є дві панелі, одна з результатом роботи, інша з такими функціями як:

– завантаження зображення;

– поріг яскравості контурів, який вмикається за бажанням користувача.

Для того щоб розпочати ДК об’єктів на зображенні, необхідно завантажити зображення натиснувши на кнопку «Загрузить изображение», після чого автоматично з’явиться результат роботи кожного оператора.

4.4 Тестування розробленої моделі

4.4.1 Тестування програми

Натискаючи на кнопку для завантаження зображення, треба обрати зображення для редагування та в результаті чого буде отримано наступний вигляд програми (рис. 4.6, рис. 4.7).

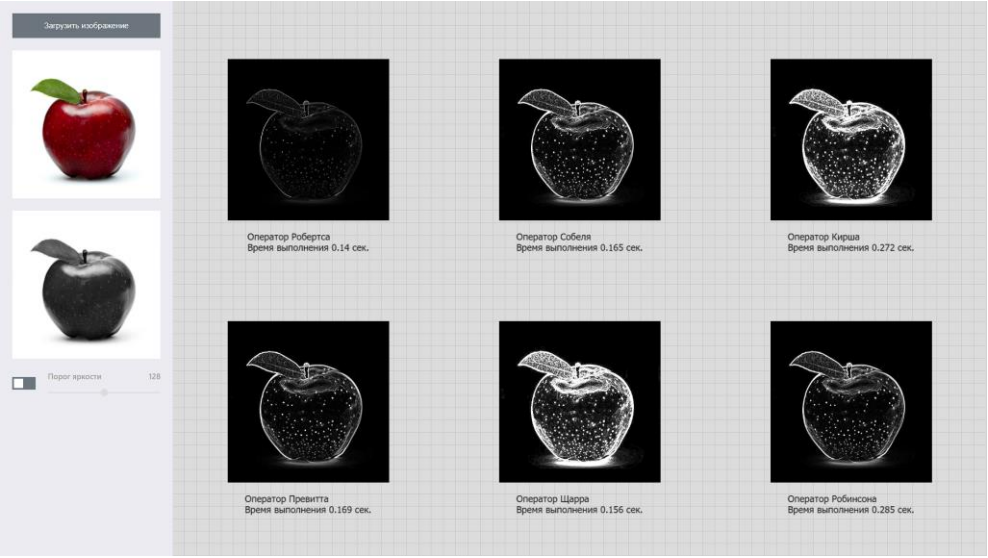

Рисунок 4.6 – Загальний результат роботи програми без використання медіанного фільтру

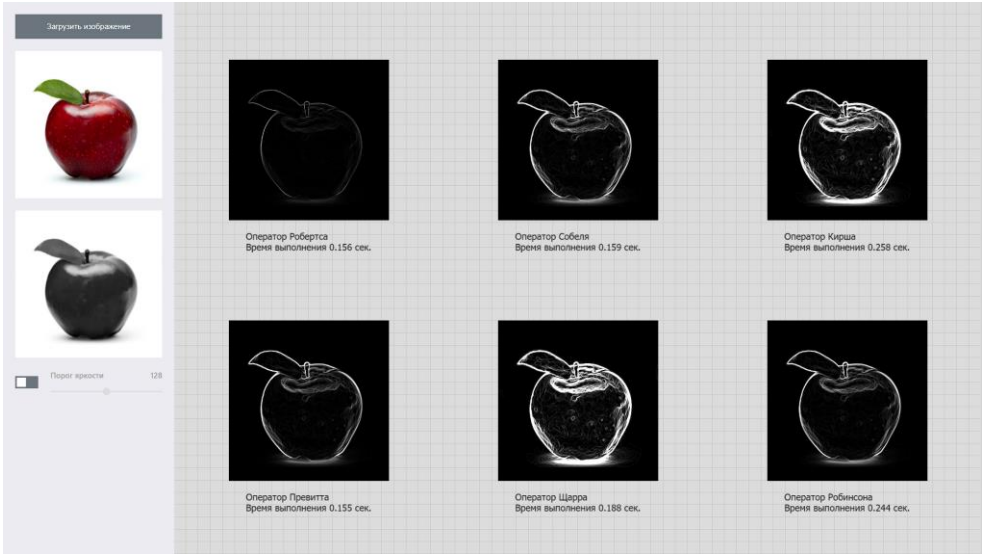

Рисунок 4.7 – Загальний результат роботи програми з використанням медіанного фільтру

На панелі з вхідним зображенням можна спостерігати як змінилося зображення, а саме було переведено до градації сірого та виконане видалення шумів (рис. 4.8, рис. 4.9).

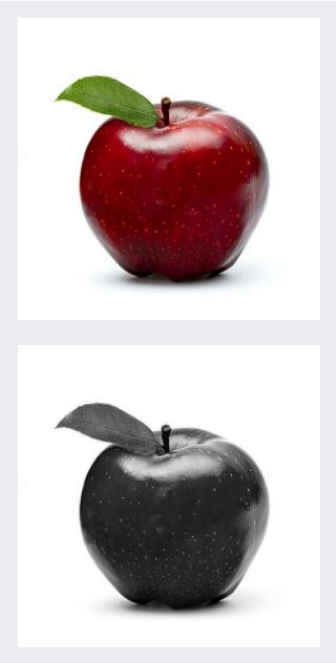

Рисунок 4.8 – Перетворення вхідного зображення без використання медіанного фільтру

Рисунок 4.9 – Перетворення вхідного зображення з використанням медіанного фільтру

Як можна спостерігати, білі крапки на яблуці стали майже непомітними, поверхня яблука стала немов глянцева, що істотно зіграє роль у подальшому результаті ДК об’єктів на зображенні.

Результат роботи операторів та час виконання кожного з них продемонстровано нижче (рис. 4.10, рис. 4.11).

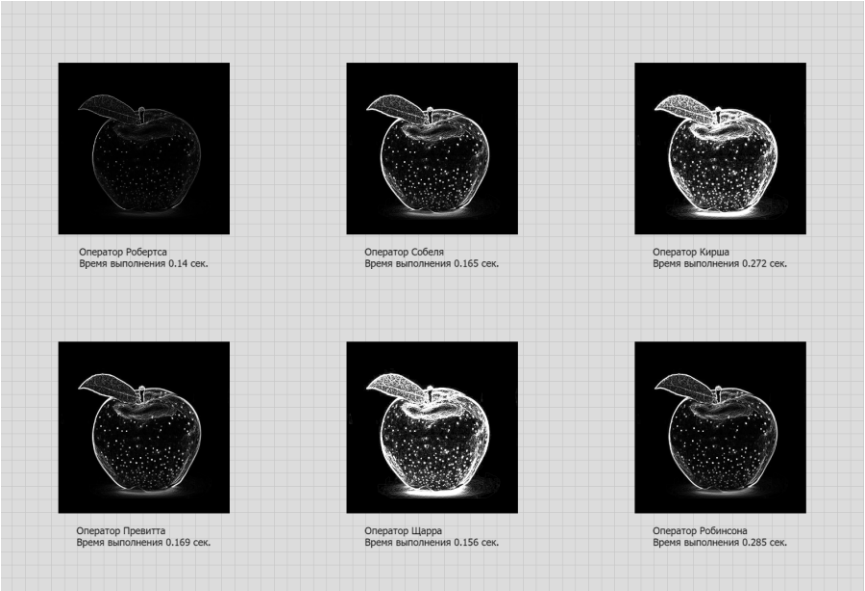

Рисунок 4.10 – Результат роботи операторів ІК без використання медіанного фільтру

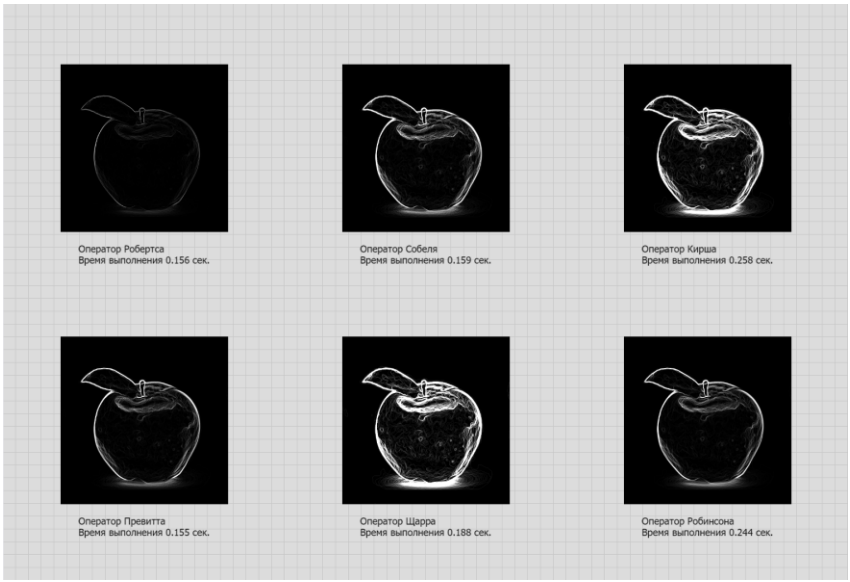

Рисунок 4.11 – Результат роботи операторів ІК з використанням медіанного фільтру

Для того щоб зменшити кількість інформації, а саме видалити немаксимуми, можна скористатись функцією порогу яскравості яка має вигляд (рис. 4.12).

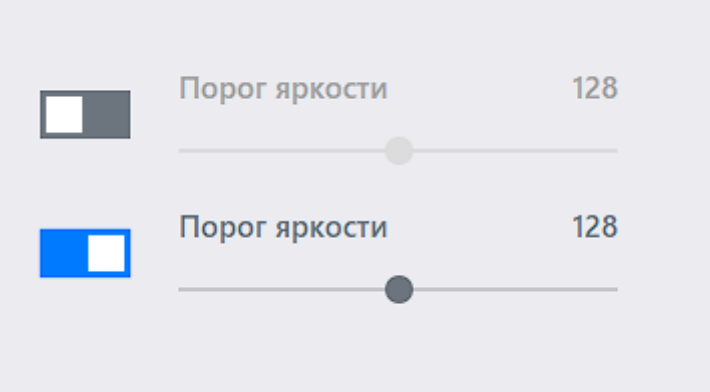

Рисунок 4.12 – Результат роботи операторів ІК

Після видалення немаксимумів маємо наступний результат (рис. 4.13 – 4.16).

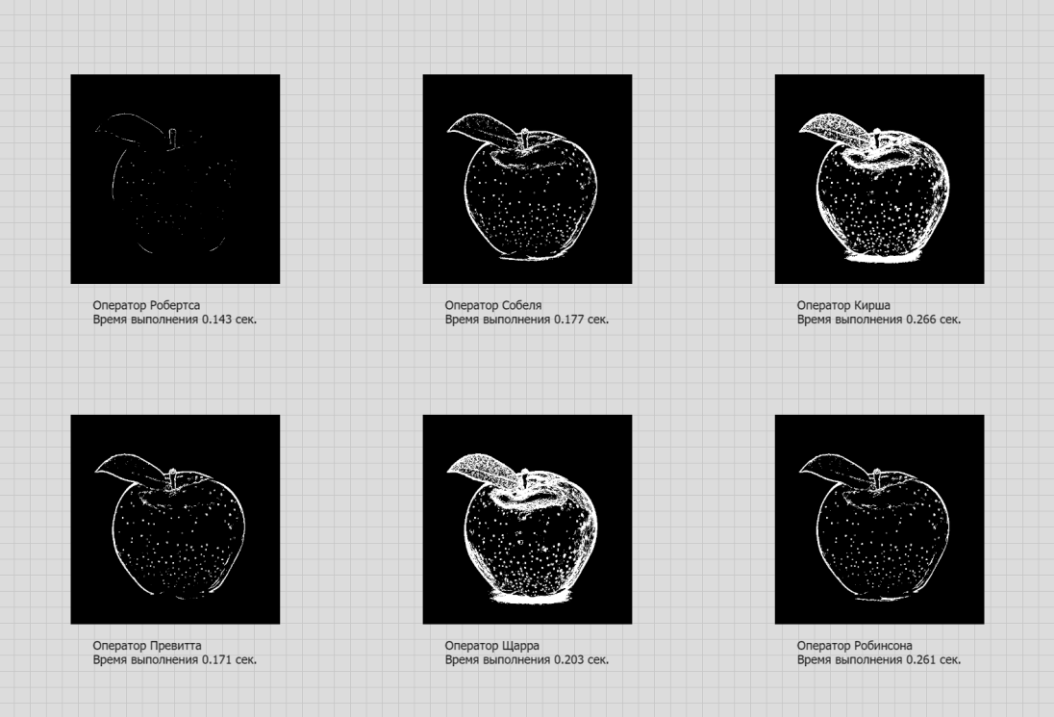

Рисунок 4.13 – Результат роботи видалення немаксимумів при 128 значенні без використання медіанного фільтру

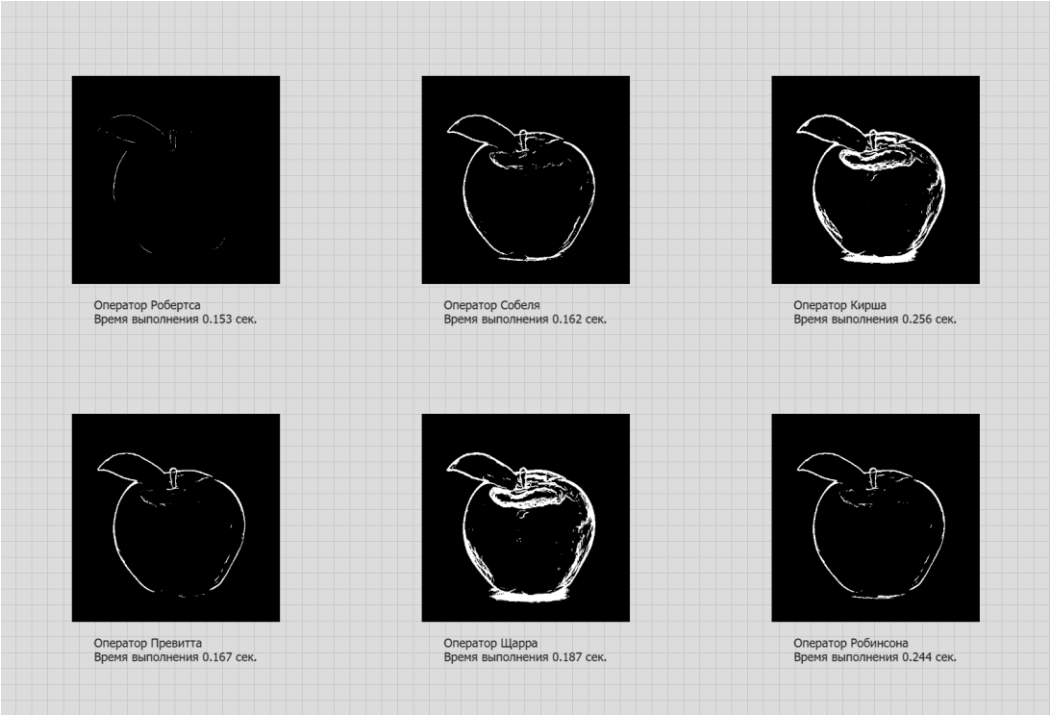

Рисунок 4.14 – Результат роботи видалення немаксимумів при 128 значенні з використанням медіанного фільтру

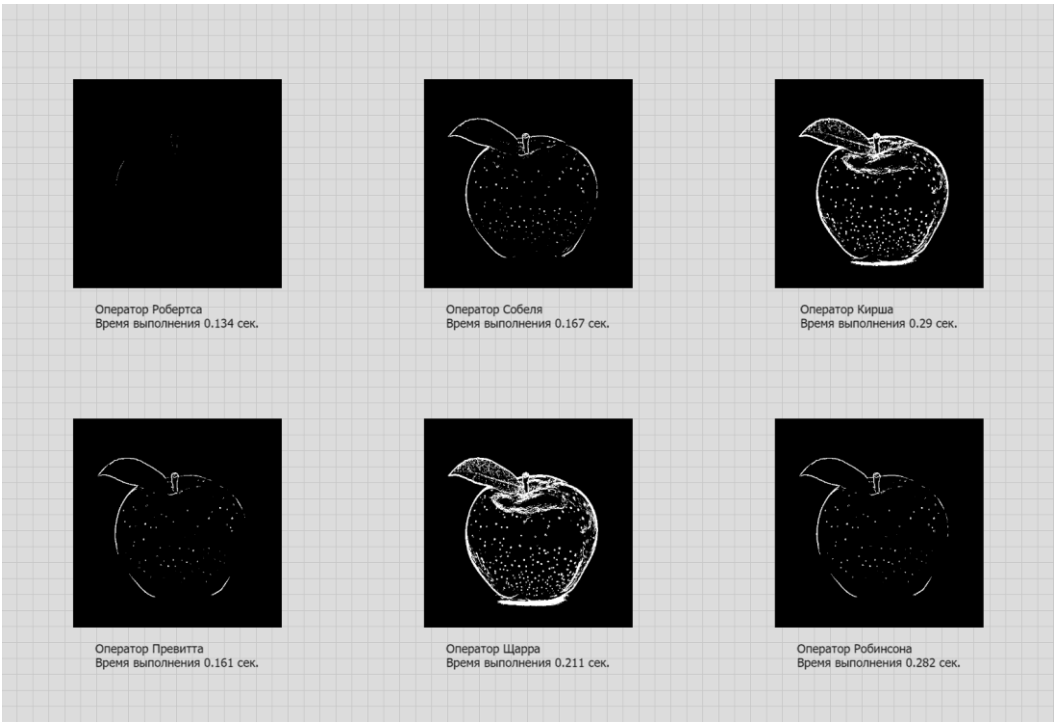

Рисунок 4.15 – Результат роботи видалення немаксимумів при 255 значенні без використання медіанного фільтру

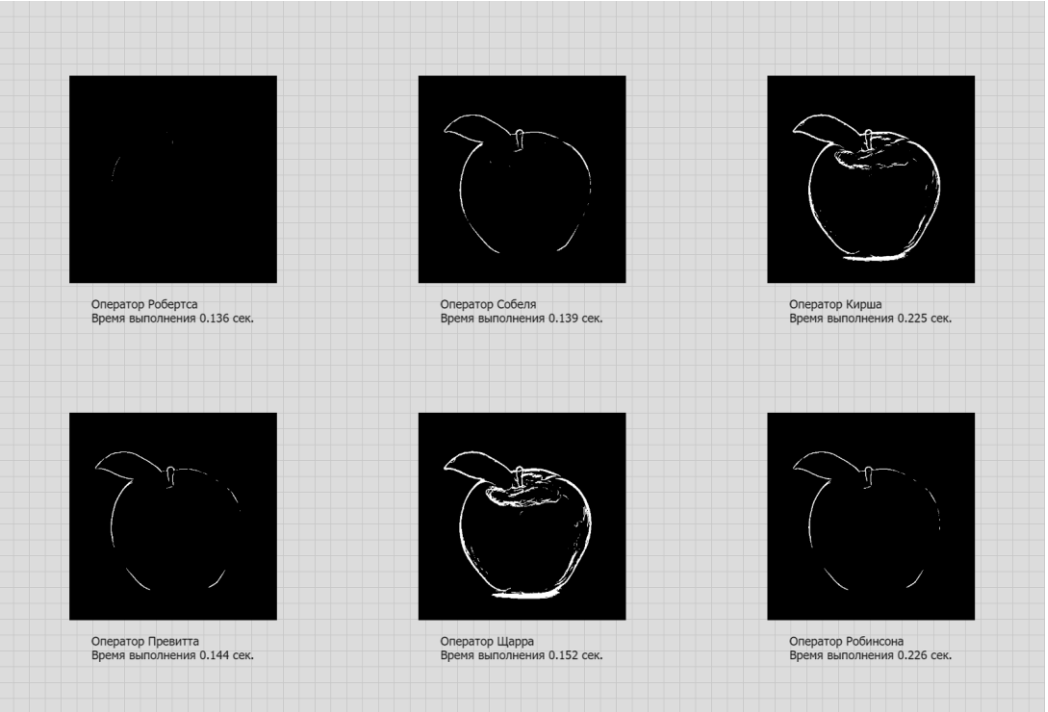

Рисунок 4.16 – Результат роботи видалення немаксимумів при 255 значенні з використанням медіанного фільтру

4.4.2 Аналіз роботи програми

Можна зробити висновки щодо роботи алгоритмів, а саме кажучи про швидкість та якість візуальної інформації.

Таблиця 4.1 – Аналіз роботи операторів ІК

Оператор	**Середня швидкість (секунди)**	**Візуальна якість**
Робертса	0.144	Мало інформації
Превітта	0.156	Задовільно
Собеля	0.152	Задовільно
Шарра	0.168	Забагато інформації
Кірша	0.256	Забагато інформації
Робінсона	0.248	Задовільно

За результатами таблиці можна зауважити, що по швидкості оператори Робертса, Превітта, Собеля та Шарра працюють швидше ніж оператори Кірша та Робінсона. Незважаючи на те, що оператор Робертса має ядро 2×2 та меншу швидкість числення, завдяки сучасним ЕВМ робота пізніх версій алгоритмів знаходиться на одному рівні з ним, але якість цього алгоритму не є задовільною.

Найдовшими в плані швидкості є оператори Кірша та Робінсона із-за їх 8-ми напрямків по яким крутять маски.

Кажучи про візуальну складову, найкраще всього показали себе оператори Превітта, Собеля та Робінсона.

ВИСНОВКИ

У рамках атестаційної роботи було розроблено і реалізовано алгоритми для ідентифікації контурів об’єктів на зображенні. Проведено аналіз та порівняння роботи існуючих операторів завдяки якому можна зробити висновки щодо вибору алгоритму для подальшого використання. За допомогою модифікації алгоритмів у вигляді видалення шумів було отримано набагато кращі результати.

Найбільш приближені результати показали оператори Собеля та Превітта. Вони є оптимальним варіантом між швидкістю та якістю ідентифікації контурів. Найшвидшим є перехресний оператор Робертса завдяки його маленькому ядру, але якість ІК не є задовільною. Найповільнішими операторами є оператори Кірша та Робінсона через їх кількість ядер.

Були проаналізовані усі проблеми, недоліки та достатки більшої частини алгоритмів детектування контурів об’єктів на зображенні та зроблені висновки щодо них.

Практична значущість роботи – підвищення якості детектування контурів, видалення шумів та отримання прикладних програмних моделей для дослідження та впровадження методів класифікації в системах комп’ютерного зору при використанні розпізнавання образів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Гонсалес, Р., & Вудс, Р. (2012). Цифровая обработка изображений.

2. Грузман, И. С., Киричук, В. С., Косых, В. П., Перетягин, Г. И., & Спектор, А. А. (2002). Цифровая обработка изображений в информационных системах. Новосибирск: Изд-во НГТУ, 352, 6.

3. Абламейко, С. В., & Лагуновский, Д. М. (2000). Обработка изображений: технология, методы, применение.

4. Гашников, М. В., Глумов, Н. И., Ильясова, Н. Ю., Мясников, В. В., Попов, С. Б., Сергеев, В. В., ... & Чичева, М. А. (2003). Методы компьютерной обработки изображений.

5. Бутаков, Е. А., Островский, В. И., & Фадеев, И. Л. (1987). Обработка изображений на ЭВМ. Радио и связь.

6. Визильтер, Ю. В., Желтов, С. Ю., Князь, В. А., Ходарев, А. Н., & Моржин, А. В. (2012). Обработка и анализ цифровых изображений с примерами на LabVIEW и IMAQ Vision.

7. Canny, J. (1986). A computational approach to edge detection. IEEE Transactions on pattern analysis and machine intelligence, (6), 679-698.

8. Marr, D., & Hildreth, E. (1980). Theory of edge detection. Proceedings of the Royal Society of London. Series B. Biological Sciences, 207(1167), 187-217.

9. Ziou, D., & Tabbone, S. (1998). Edge detection techniques-an overview. Pattern Recognition and Image Analysis C/C of Raspoznavaniye Obrazov I Analiz Izobrazhenii, 8, 537-559.

10. Нестеров, А. Ю., & Белов, Ю. С. (2016). Распознавание образов по уникальным точкам на примере дорожных знаков. *Электронный журнал: наука, техника и образование*, (4), 113-119.

11. Perona, P., & Malik, J. (1990). Scale-space and edge detection using anisotropic diffusion. IEEE Transactions on pattern analysis and machine intelligence, 12(7), 629-639.

12. Митропольский, Н. Н., & Ковшов, Е. Е. (2008). Сегментация и идентификация контуров объектов на цифровых изображениях. Известия высших учебных заведений. Проблемы полиграфии и издательского дела, (5), 019-028.

13. Бархатов, В. А. (2006). Обнаружение сигналов и их классификация с помощью распознавания образов. *Дефектоскопия*, *4*, 14-27.

14. Білинський, Й. Й., Огородник, К. В., & Микулка, І. В. (2012). Детектор виділення контуру розфокусованих зображень. Наукові праці Вінницького національного технічного університету, (3).

15. Василенко, М. Ю. (2020). Методологія детектування об'єктів на зображенні (Doctoral dissertation, ВНТУ).

16. Шлезингер, М., & Главач, В. (2004). Десять лекций по статистическому и структурному распознаванию. *Киев: Наукова думка*, *545*.

17. Неснов, С. О. (2016). Приховування інформації у контурах растрових зображень (Doctoral dissertation, ВНТУ).

18. Симанков, В. С., & Луценко, Е. В. (1999). Адаптивное управление сложными системами на основе теории распознавания образов.

19. Мокшин, В. В., Сайфудинов, И. Р., Кирпичников, А. П., & Шарнин, Л. М. (2016). Распознавание образов транспортных средств на основе эвристических данных и машинного обучения. Вестник Казанского технологического университета, 19 (5).

20. Борисенко, В. И., Пионтковский, Ю. Ф., & Горанчук, В. В. (2016). Модель формирования сигнала внутризонного детектора нейтронов. *Ядерна фізика та енергетика*, (17,№ 4), 364-373.

21. Любчик, В. Р., Клепіковський, А. В., & Ковалев, В. О. (2016). Методика проектування комплексів оптико-телевізійного наведення з використанням стереоскопічних оптосистем.

22. Торхов, Д. И., & Еланцев, М. О. (2016). Выбор контурного детектора в задачах аэронавигации по зрительным образам. In *Молодые ученые-ускорению научно-технического прогресса в XXI веке* (pp. 432-437).

23. Жуков, О. А. МЕТОД ВИДІЛЕННЯ КОНТУРІВ НА ЦИФРОВИХ ЗОБРАЖЕННЯХ. *Інфокомунікації–сучасність та майбутнє: матеріали п'ятої міжнар.*, 30.

24. Мазуров, В. Д. (2010). Математические методы распознавания образов.

25. Аркадьев, А. Г. (2013). *Обучение машины распознаванию образов*. Рипол Классик.

26. Симанков, В. С., & Луценко, Е. В. (1999). Адаптивное управление сложными системами на основе теории распознавания образов.

27. Татьянкин, В. М. (2016). Подход к формированию архитектуры нейронной сети для распознавания образов. *Вестник Югорского государственного университета*, (2 (41)).

28. Чабан, Л. Н. (2004). Теория и алгоритмы распознавания образов. *М.: МИИГАиК*, 26-29.

29. Гук, А. П. (2015). Автоматизация дешифрирования снимков. Теоретические аспекты статистического распознавания образов. *Известия высших учебных заведений. Геодезия и аэрофотосъемка*, (S5), 166-169.

30. Степанян, И. В., & Зиеп, Н. Н. (2018). Растущие сверточные нейроподобные структуры для задач распознавания статических образов. *Нейрокомпьютеры: разработка, применение*, (5), 4-11.

31. Оганезов, А. (2006). *Применение нейронных сетей в задачах распознавания образов* (Doctoral dissertation).

32. Arbelaez, P., Maire, M., Fowlkes, C., & Malik, J. (2010). Contour detection and hierarchical image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, *33*(5), 898-916.

33. Grigorescu, C., Petkov, N., & Westenberg, M. A. (2003). Contour detection based on nonclassical receptive field inhibition. *IEEE Transactions on image processing*, *12*(7), 729-739.

34. Papari, G., & Petkov, N. (2011). Edge and line oriented contour detection: State of the art. *Image and Vision Computing*, *29*(2-3), 79-103.

35. Martelli, A. (1976). An application of heuristic search methods to edge and contour detection. *Communications of the ACM*, *19*(2), 73-83.

36. Brigger, P., Hoeg, J., & Unser, M. (2000). B-spline snakes: a flexible tool for parametric contour detection. *IEEE Transactions on image processing*, *9*(9), 1484-1496.

37. Xiaofeng, R., & Bo, L. (2012). Discriminatively trained sparse code gradients for contour detection. In *Advances in neural information processing systems* (pp. 584-592).

38. Guttman, M. A., Prince, J. L., & McVeigh, E. R. (1994). Tag and contour detection in tagged MR images of the left ventricle. *IEEE transactions on Medical Imaging*, *13*(1), 74-88.

39. Lim, J. J., Zitnick, C. L., & Dollár, P. (2013). Sketch tokens: A learned mid-level representation for contour and object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 3158-3165).

40. Bertasius, G., Shi, J., & Torresani, L. (2015). Deepedge: A multi-scale bifurcated deep network for top-down contour detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4380-4389).